UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA DE PRODUÇÃO

METODOLOGIA PARA O GERENCIAMENTO DISTRIBUÍDO DE PROJETOS E MÉTRICA DE SOFTWARE

Fabiane Barreto Vavassori

Fernando Álvaro Ostuni Gauthier, Dr.
Orientador

Florianópolis(SC), julho de 2002

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

METODOLOGIA PARA O GERENCIAMENTO DISTRIBUÍDO DE PROJETOS E MÉTRICA DE SOFTWARE

Fabiane Barreto Vavassori

Tese apresentada ao
Programa de Pós-Graduação em
Engenharia de Produção da
Universidade Federal de Santa Catarina
como requisito parcial para obtenção
do grau de Doutora em
Engenharia de Produção

Florianópolis

2002

A minha dinda, Maria do Carmo, pelo apoio e carinho dedicados a mim quando ainda dava meus primeiros passos na graduação.

Agradecimentos

Esta tese só se tornou possível pela grandiosidade da amizade que encontrei ...

... em meu orientador prof. Dr. Fernando Gauthier, fonte de tranqüilidade, confiança e sabedoria prestada até a conclusão deste trabalho.

... em minhas amigas especiais Ana, Anita e Betinha, vocês nunca me deixaram desanimar... foram sempre exemplo de dedicação e perseverança.

... aos meus colegas de trabalho que compartilharam comigo seus conhecimentos, muitas vezes contribuindo, até mesmo sem saber, para alcançar os resultados deste trabalho.

... nos momentos em que ensinei e também aprendi com os meus "filhos" – Everton, Júlio, Kokubo e Jonas. Foram momentos que propiciaram muita troca e aprendizagem mútua. Mas, em especial, ressalto uma gratidão especial ao Everton e ao Júlio, que desempenharam papel fundamental na realização deste trabalho.

... nos membros da banca examinadora: Prof Marco Antônio Barbosa Cândido, Prof[a]. Angelita Maria de Ré, Prof[a]. Anita Maria da Rocha Fernandes, Prof. João Bosco Mangueira Sobral, Prof. Alejandro Martins Rodriguez e também ao Prof. Roberto Pacheco os quais prestaram valiosas sugestões durante o *qualify* e na versão final desta tese.

... na direção do Centro de Ciências Tecnológicas da Terra e do Mar (CTTMar), Prof. Fernando Diehl, e coordenação do curso de Ciência da Computação, Prof. Luis Carlos Martins, da Universidade do Vale do Itajaí, pelo apoio e atenção dispensada em todos os momentos em que foram solicitados.

...em meu noivo, Alexandre, que acompanhou toda esta caminhada sempre ao meu lado.

...e, em especial, a minha família (que é a minha base) que com incansável dedicação, sempre estiveram presentes.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

AIE	Arquivos de Interface Externa
ALI	Arquivos Lógicos Internos
BFPUG	Brazilian Function Point Users Group
CASE	Computer-Aided Software Engineering
CE	Consultas Externas
CGS	Características Gerais dos Sistemas
CMM	Capability Maturity Model
COCOMO	Constructive Cost Model
CPM	Critical Path Method
CPU	Central Processing Unit
CSCW	Computer-Supported Cooperative Work
DER	Dados Elementares Referenciados
DE-R	Diagrama Entidade-Relacionamento
EE	Entradas Externas
EF	Environment Factor
ES	Engenharia de Software
FPA	Function Point Analysis
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	HyperText Markup Language
IA	Inteligência Artificial
ISO	International Organization for Standardization
IFPUG	International Function Point Users Group
IP	Internet Protocol
JDBC	Java Database Connectivity
LOC	Lines of Code
MIPS	Número de Instruções por Segundo
NI	Nível de Influência
NIT	Nível de Influência Total
PERT	Program Evaluation and Review Technology
PF	Pontos de Função
PFA	Pontos por Função de Aplicações Instaladas

PFD	Pontos por Função de Projetos de Desenvolvimento
PFM	Pontos por Função de Projetos de Manutenção
PMI	Project Management Institute
RBC	Raciocínio Baseado em Casos
RLR	Registros Lógicos Referenciados
RMI	Remote Method Invocation
SE	Saídas Externas
SLOC	Source Line of Code
TCF	Technical Complexity Factor
UML	Unified Modeling Language
UAW	Unajusted Actor Weights
UCP	Use Case Points
UUCW	Unajusted Use Case Weights
UUPC	Unadjusted Use Case Points

RESUMO

Vavassori, Fabiane Barreto. **Metodologia para o Gerenciamento Distribuído de Projetos e Métrica de Software.** 2002. Tese (Doutorado em Engenharia de Produção) – UFSC, Florianópolis, SC.

Diversos são os problemas relacionados ao desenvolvimento de software, os quais são resultantes da omissão ou do mau uso de metodologias e técnicas adequadas a esta importante tarefa de engenharia. No entanto, boa parte dos fracassos no que diz respeito aos projetos de software deve-se, principalmente, a problemas de administração ou gerenciamento do desenvolvimento de software. Sendo, portanto, o gerenciamento de projetos de software uma tarefa de fundamental importância no processo de desenvolvimento de um produto. Tendo em vista esta realidade, este trabalho apresenta uma metodologia e uma ferramenta CASE de apoio que tem como objetivos: (i) viabilizar o planejamento e gerenciamento de projeto considerando a integração com outros projetos; (ii) viabilizar a definição uma estrutura de divisão de trabalho e, (iii) gerar para o gerente de projeto, úteis estimativas de esforço, custo e duração de um projeto de software. Além disso, o gerente pode usar a ferramenta para compilar métrica de software, que por fim oferecerá uma indicação da produtividade no desenvolvimento de software e da qualidade do produto. Esta metodologia é proposta para atuar em projetos cujo gerenciamento se dá de forma distribuída, uma vez que se observou esta ser uma prática que vem ocorrendo significantemente e não há pesquisas, metodologias ou ferramentas que dêem suporte a esta atividade sobre o enfoque tratado neste trabalho. Além disto, é agregada métrica de software para auxiliar o gerenciamento, característica esta que também não é comumente englobada pelas ferramentas de gerenciamento. Desta forma, este trabalho apresenta uma metodologia suportada por uma ferramenta CASE de apoio, a qual emprega a tecnologia de agentes, para viabilizar o planejamento e o gerenciamento distribuído de projetos agregando métrica de software.

Palavras-Chave: [gerenciamento de projeto] [CASE] [Métrica de Software]

ABSTRACT

VAVASSORI, Fabiane Barreto. **Methodology for Distributed Management of Projects and Software Metric.** 2002. Thesis (Doctor Degree in Production Engineering) - UFSC, Florianópolis, SC.

There are several problems related to the software development, which are results of the omission or the bad use of methodologies and appropriate techniques for this important engineering task. However, a good part of the failures with regard to software projects is due, mainly, to administration problems or management of software development. Therefore, the management of software projects is a task of fundamental importance in the process of product development. From this reality, this paper presents a methodology and a CASE tool of support that has as goals: (i) to enable the project planning and management considering the integration with other projects; (ii) to make possible the definition of a structure of work division and, (iii) to generate for the project manager, useful estimates of effort, cost and duration of a software project. Besides, the manager could use the tool to compile the software metric, which finally will offer an indication of the productivity in the software development and of the product quality. This methodology is proposed to act in projects whose management happens in a distributed way, once in this was observed to be a practice that is happening significantly and there are no researches, methodologies or tools that give support to this activity this is the focus studied in this paper. Besides, the software metric to aid the management is included, a feature that is not also included commonly by management tools. So, this paper presents a methodology supported by a CASE tool of support, which uses the agents' technology to make possible the planning and the distributed management of projects joining software metric.

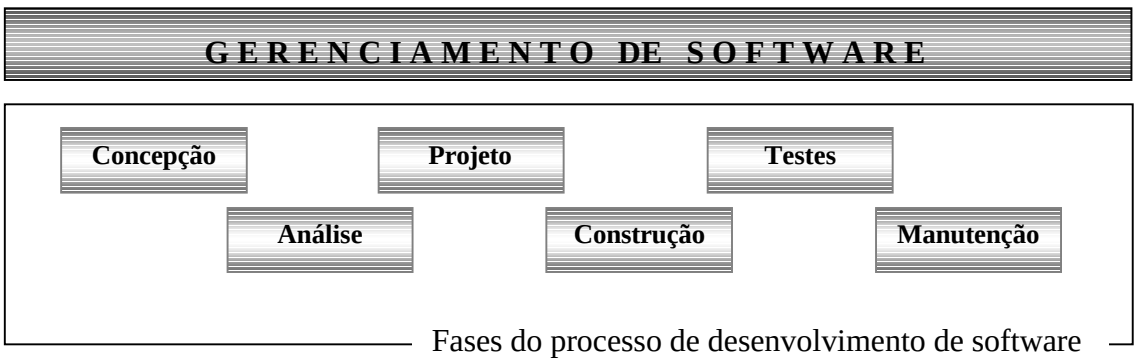
Keywords: [Project Management] [CASE] [Software Metric]

1 INTRODUÇÃO

1.1 Contextualização

O presente trabalho situa-se na sub-área de Gerenciamento de Projeto de Software, assunto este tratado dentro da área de Engenharia de Software. O gerenciamento constitui-se em uma tarefa de fundamental importância no processo de desenvolvimento de um produto, definido como uma primeira camada deste processo. O gerenciamento de projeto não é visto como uma etapa clássica do processo de desenvolvimento, uma vez que ele acompanha a todas as etapas, da concepção à manutenção do produto, conforme retratado na Figura 1.

Figura 1: Área em Estudo



Para que um projeto de software seja bem sucedido, é necessário que alguns parâmetros sejam corretamente analisados, como por exemplo, o escopo do software, os riscos envolvidos, os recursos necessários, as tarefas a serem realizadas, os marcos de referência a serem acompanhados, os esforços e custos aplicados e a sistemática a ser seguida. A análise de todos estes parâmetros, segundo Pressman (1995), é a função típica do gerenciamento de projetos. Função esta, que se inicia antes do trabalho técnico e que prossegue à medida que o software vai se concretizando na forma de um produto.

1.2 Apresentação do Problema

De acordo com Capers Jones (*apud* Chang & Christensen 1999) "a maioria dos esforços em engenharia de software tem se preocupado em construir ferramentas CASE para auxiliar no projeto, implementação e teste, enquanto os métodos formais e ferramentas usadas para medir, planejar, estimar e monitorar os projetos de software são praticamente inexistentes".

Atualmente esta afirmação ainda é válida, pois se pode perceber que há um grande número de ferramentas para as fases de desenvolvimento quando comparado com as ferramentas de gerenciamento de projeto. Observação esta comprovada ao se realizar o levantamento de requisitos, onde se buscou por ferramentas de gerenciamento de projetos e métricas de software visando identificar as suas características. Este estudo resultou em uma análise comparativa (descrita nas seções, 3.4, 3.5 e 4.5) entre 6 ferramentas de gerenciamento de projetos, 4 ferramentas de métricas de software e 4 projetos de pesquisas envolvendo ferramentas para gerenciamento de projetos, listados no Quadro 1.

Quadro 1: Projetos e Ferramentas Analisadas

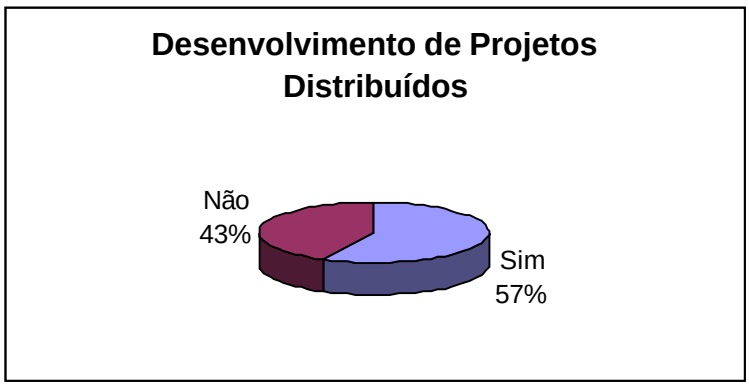
Projetos de Pesquisa	**Ferramentas de gerenciamento**	**Ferramentas de métricas**
• Prompter (O'Connor, 2000) • CSCW + gerenciamento de projeto (Knotts et al. 1998) • DSPMtool (Surjaputra, R. & Maheshwari, 1999) • SPPA (Wu & Simmons 2000)	• FastTrack • Task Manager • Delegator • Alexys Team • MS-Project • Super Project	• Costar • USC-COCOMO • Calico • Cost Xpert 2.1

Notou-se, conforme afirmado por Capes Jones (*apud* Chang & Christensen 1999) que, dentre os softwares analisados, nenhum viabiliza o acompanhamento de um projeto agregando em uma mesma ferramenta o emprego de métricas de software. Assim, este projeto visa integrar métricas de software à metodologia proposta, pois se acredita que com esta integração pode-se gerar informações que fundamentem de forma mais precisa a tomada de decisão por parte dos gerentes de projeto.

Outro aspecto observado nesta pesquisa refere-se ao gerenciamento de projetos distribuídos. Observou-se que dentre as ferramentas e projetos envolvendo aspectos de gerenciamento apenas algumas ferramentas abordam superficialmente o gerenciamento de projetos desenvolvidos de forma não centralizada.

No entanto, durante o levantamento para estabelecimento das características pertinentes ao gerenciamento de projetos foi elaborada uma pesquisa exploratória onde foram aplicados questionários (disponibilizado no Anexo I) em empresas de desenvolvimento de software das cidades de Florianópolis, Blumenau, Itajaí e Balneário Camboriú. Neste levantamento buscava-se entrar em contato com a realidade empregada pelos gerentes de projetos. Um dos resultados obtidos demonstra que, conforme pode ser visualizado no gráfico da Figura 2, 57% das empresas entrevistadas já desenvolveram algum projeto distribuído .

Figura 2: Gráfico Relativo ao Desenvolvimento de Projetos Distribuídos



Ou seja, confrontando esta realidade com o levantamento comparativo das ferramentas, verificou-se que apenas 2 ferramentas analisadas possuem algum suporte ao compartilhamento de informações. Sendo, no entanto, este suporte viabilizado através de publicação de resultados na Web para acompanhamento da equipe ou compartilhamento da base de dados pela equipe. Ambas ferramentas não abordam este aspecto em profundidade e não suportam o planejamento distribuído de projeto. Também foram analisados projetos de pesquisa que tem como foco o gerenciamento/planejamento de projetos, tais projetos (descritos na seção 3.4) utilizam

tecnologias como agentes e *groupware*[1] para atuar no gerenciamento. De maneira geral, as pesquisas visam o aprimoramento no suporte a tomada de decisão, utilizando arquitetura distribuída, contudo, com enfoques diferenciados do abordado neste trabalho.

Desta forma, considerando que o enfoque central deste trabalho é disponibilizar aos gerentes de projeto uma visão global que permita o acompanhamento de diversos projetos correlacionados e, tendo-se observado que os projetos de pesquisas não enfocam este aspecto e as ferramentas disponíveis não tratam estes casos, considerou-se este um amplo campo a ser explorado.

1.3 Pergunta de Pesquisa

As perguntas de pesquisa que surgiram a partir do problema em questão refere-se a "Como gerenciar e planejar projetos distribuídos de software?" e "Como integrar o emprego de métrica de software ao gerenciamento/planejamento de projeto de software?"

1.4 Objetivos

1.4.1 Objetivo Geral

Desenvolver uma metodologia para viabilizar o planejamento e gerenciamento distribuído de projetos e integrar uma métrica de software para suporte a decisão.

Para um perfeito entendimento do objetivo geral desta proposta faz-se necessário a definição exata do propósito ao qual ela se destina. Para tanto, deve ser analisado mais detalhadamente o termo: "gerenciamento distribuído de projetos". Isto implica em considerar um único produto obtido a partir de vários projetos. Ou seja:

tem-se diferentes partes trabalhando juntas por algum tempo visando alcançar um objetivo comum.

---

[1] No campo do CSCW (*Computer-Supported Cooperative Work*), *groupware* se refere ao software usado para suportar a comunicação e coordenação das necessidades das pessoas para trabalhar em grupo.

Cada trecho desta definição foi cuidadosamente escolhido para focar a discussão em um tipo específico de projeto:

- *diferentes partes* – as pessoas envolvidas no projeto são de diferentes organizações (ou não se encontram em uma mesma sede). Projetos multi-organizacionais são especialmente desafiadores para se gerenciar, pois cada organização introduz seus próprios objetivos, cultura, processos, dentre outros.
- *trabalhando juntas* – são gastos esforços no intuito de manter a cooperação entre as partes.
- *por algum tempo* – os projetos possuem um *deadline*. E, geralmente, o não cumprimento de datas limites por uma das partes, interfere diretamente no planejamento das demais partes envolvidas.
- *visando alcançar um objetivo comum* – existe uma razão pela qual as partes estão trabalhando juntas – o desenvolvimento de um produto/aplicação específico. A produção deste produto está atrelada ao gerenciamento de todos os projetos correlacionados.

Portanto, é dentro deste escopo que a proposta está focada. Visando auxiliar em aspectos pertinentes ao planejamento e gerenciamento distribuído de projetos, tendo sempre em vista que os projetos são correlacionados. E, sobre esta concepção é importante perceber que alguma alteração efetuada em qualquer um dos projetos correlacionados deve influenciar de alguma forma os demais.

1.4.2 Objetivos Específicos

Dentre os objetivos específicos encontram-se:

§ Definir uma metodologia para viabilizar o planejamento e gerenciamento distribuído de projeto.

§ Integrar métrica de software ao gerenciamento de projetos, visando fornecer informações para suporte a tomada de decisão.

§ Implementar uma ferramenta CASE que suporte a metodologia elaborada.

1.5 Justificativa

> "Apesar de toda evolução tecnológica (ambientes mais produtivos), processual (processos mais bem-definidos), profissional (profissionais mais bem-formados) e do ambiente (ferramentas e equipamentos), a área de informática ainda entrega projetos fora do prazo e que não refletem a necessidade do usuário em termos de função, qualidade, prazos e custos. Sistematicamente os projetos atrasam e as estimativas não são concretizadas" (Braga, 1996).

Conforme verificado por Braga (1996), apesar da evolução da engenharia de software a área de gerenciamento de projetos de software ainda apresenta deficiências. Considerando-se que os problemas aumentam quando adicionado o fator do projeto ser desenvolvido de forma distribuída e que as pesquisas e ferramentas atualmente existentes não oferecem apoio efetivo ao monitoramento de projetos com esta característica tem-se, assim, um importante campo de pesquisa.

Outro ponto a ser considerado parte da afirmativa também feita por Braga (1996) de que "não se pode gerenciar o que não se pode medir". É importante estar ciente que as medidas são uma forma para se estimar prazos, custos e avaliar a produtividade do desenvolvimento de software. Desta forma, torna-se importante integrar a métrica de software ao planejamento/gerenciamento de projetos, como forma de viabilizar informações consistentes para a tomada de decisão pertinente ao gerenciamento de projeto.

Um benefício evidente desta integração é citado por Pressman (1995) onde o autor afirma que sem informações quantitativas a respeito de todo o processo de desenvolvimento de softwares por parte de uma empresa ou equipe de desenvolvedores de produto, é impossível tirar qualquer conclusão sobre de que forma está evoluindo a produtividade. Assim, através da obtenção de medidas relativas à produtividade e à qualidade, é possível que metas de melhorias no processo de desenvolvimento sejam estabelecidas. A obtenção de informações quantitativas é feita através do emprego sistemático de métricas de software e, para associá-las ao processo de desenvolvimento faz-se necessário a presença também de um processo de gerenciamento bem definido.

Ou seja, a avaliação quantitativa do desenvolvimento viabiliza promover pequenos ajustes no processo de desenvolvimento como forma de eliminar ou reduzir as causas de problemas que afetam de forma significativa o projeto de software.

Todos os pontos abordados até o presente momento se referem a projetos elaborados por uma única equipe. No entanto, é importante salientar que em projetos distribuídos fazem-se necessárias informações relativas tanto ao planejamento de cada projeto correlato individualmente, quanto informações relativas ao projeto como um todo. Pois, o sucesso do desenvolvimento está atrelado não só ao gerenciamento de cada parte mas também, ao gerenciamento do projeto geral. Tem-se, portanto, com esta forma de desenvolvimento, todos os problemas pertinentes a um projeto individual acrescido ainda dos problemas relativos à integração entre as gerências de cada projeto. Tópico este pouco abordado na literatura e desconsiderado pelas ferramentas de auxílio existentes atualmente.

1.6 Estrutura do Trabalho

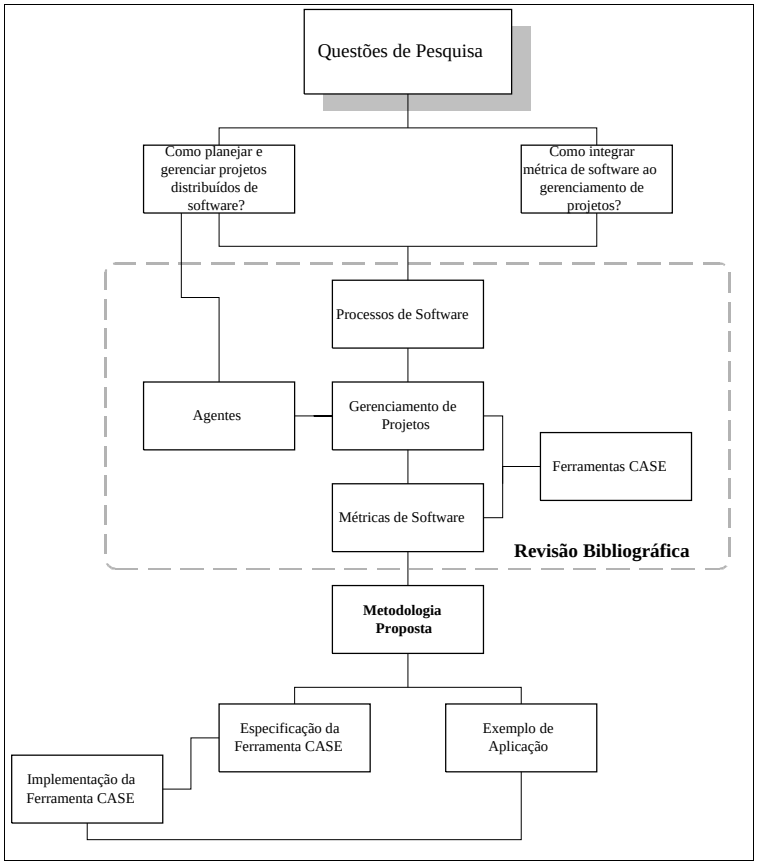
Este documento foi estruturado tendo em vista a estrutura exposta na Figura 3. abordando 10 capítulos, onde cada um constitui uma parte essencial do estudo feito para a elaboração deste trabalho.

A partir da problemática exposta nesta introdução, evidencia-se a necessidade de se ter uma metodologia para gerenciamento distribuído de projetos agregando os benefícios da utilização de métricas de software.

Uma vez contextualizada a proposta, o capítulo 2 versa sobre aspectos relevantes ao processo de software considerado uma das principais abordagens da engenharia de software. Este capítulo apresenta a relação entre processo de gerenciamento e processo de desenvolvimento de software, incluindo também a descrição dos principais modelos de processo de desenvolvimento de software.

No capítulo 3 são detalhados diversos aspectos pertinentes ao gerenciamento de projetos de software, tais como atividades do gerente, dimensões e fases a serem consideradas no gerenciamento, ferramentas de auxílio ao gerente, bem como é apresentado um estudo de algumas ferramentas CASE e projetos de pesquisa abordando gerenciamento de projetos acompanhado de uma análise comparativa entre elas.

Figura 3: Estrutura do trabalho



O capítulo 4 traz informações básicas sobre métricas de software apresentando as métricas mais difundidas atualmente, juntamente com a apresentação de algumas ferramentas que implementam tais métricas.

Como um dos objetivos desta trabalho é o desenvolvimento de uma ferramenta CASE que suporte a metodologia proposta, os conceitos e classificações de ferramentas CASE são apresentados no capítulo 5.

No capítulo 6 é abordado alguns tópicos relativos a agentes, tais como conceitos, propriedades, tipologia e arquitetura, pois tal tecnologia foi selecionada para apoiar a implementação da metodologia para gerenciamento distribuído.

A metodologia proposta e a especificação da ferramenta CASE de apoio a metodologia são detalhadamente apresentadas no capítulo 7.

O capítulo 8 apresenta a implementação da ferramenta especificada e um exemplo de aplicação da metodologia como forma de apresentar a viabilidade de aplicação tanto da metodologia quanto da ferramenta CASE.

E, finalmente as conclusões obtidas com o desenvolvimento deste trabalho são apresentadas no capítulo 9.

Por fim, as referências bibliográficas utilizadas para embasar o presente trabalho são apresentadas, compondo o capítulo 10.

Como fonte complementar os apêndices A e B fornecem maiores detalhes a respeito da especificação da ferramenta CASE proposta. E, os anexos 1 e 2 apresentam o instrumento utilizado para obter algumas informações de embasamento do presente trabalho e também as telas implementadas na ferramenta CASE.

1.7 Publicações

Durante as pesquisas para o desenvolvimento desta tese alguns resultados intermediários foram obtidos e publicados em eventos científicos, sendo os mais relevantes:

Evento: SBES 2001 - Simpósio Brasileiro de Engenharia de Software
Título: Ferramenta CASE para gerenciamento de projetos e métricas de software
Realização: IME - Instituto Militar de Engenharia / SBC – Sociedade Brasileira de Software
Local: Rio de Janeiro - RJ – Brasil
Data: 03 à 05/10/2001

Evento: ENEGEP 2001 - XXI Encontro Nacional de Engenharia de Produção
Título: Uma ferramenta para gerenciamento distribuído de projetos
Realização: ABEPRO/UFRGS/UNIMEP/FTC/UFSC
Local: Salvador - BA – Brasil
Data: 17à 19/10/2001

Evento: ICIE 2001 - International Congress of Information Engineering
Título: Análise Comparativa de Ferramenta CASE para gerenciamento de projeto
Realização: Universidade de Buenos Aires
Local: Buenos Aires – Argentina
Data: 25 à 27/04/2001

Evento: CBCOMP 2001 - Congresso Brasileiro de Computação
Título: Integrando métrica de software ao gerenciamento de projetos
Organização: Universidade do Vale do Itajaí – UNIVALI
Local: Itajaí - SC – Brasil
Data: 20 à 24/08/2001

2 PROCESSOS DE SOFTWARE

Para Jalote (1997), o conceito de processo é o coração da engenharia de software. De acordo com Webster (*apud* Jalote, 1997), o termo processo significa "um método particular de fazer alguma coisa, geralmente envolvendo um número de passos ou operações." Desta forma, Jalote (1997) conclui que um processo de software

> "é um conjunto de atividades, ligadas por padrões de relacionamento entre elas, pelas quais se as atividades operarem corretamente e de acordo com os padrões requeridos, o resultado desejado é produzido. O resultado desejado é um software de alta qualidade a baixo custo. Obviamente, um processo que não aumenta a produção (não suporta projetos de software grandes) ou não pode produzir software com boa qualidade não é um processo adequado."

O processo que trata das questões técnicas e de gerenciamento do desenvolvimento de software é chamado processo de software. Vários tipos de atividades precisam ser executadas no desenvolvimento de software. No entanto, o projeto de desenvolvimento de software deve ter no mínimo atividades de desenvolvimento e atividades de gerenciamento do projeto. A soma destas atividades compreende o processo de software.

As principais entidades componentes da engenharia de software (projeto, processo e produto), bem como seus objetivos e relacionamentos são abordados na seção 2.1 – objetivando conceituar estes termos que são básicos para o trabalho. Na seqüência, a seção 2.2. aborda alguns modelos de processo de desenvolvimento de software. E, como a ênfase deste trabalho está nas atividades de gerenciamento, o capítulo 3 versa deste assunto mais detalhadamente.

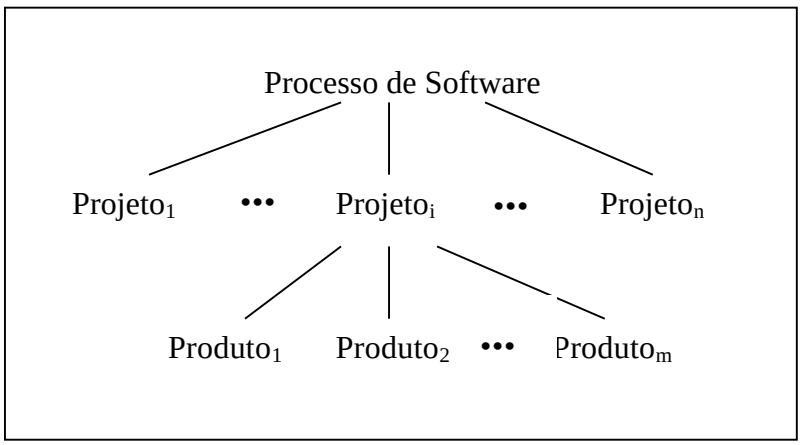
2.1 Processo, Projeto e Produto

Um processo de software, como citado anteriormente, especifica um método para o desenvolvimento de software. Um projeto de software, por outro lado, é um projeto desenvolvido no qual o processo de software é usado. E os produtos de software são os resultados do projeto de software. (Jalote, 1997)

Cada projeto de desenvolvimento de software inicia com algumas necessidades e é finalizado com o software que satisfaz estas necessidades. Um processo de software especifica o conjunto de atividades que devem ser executadas para a partir da necessidade do usuário chegar no produto final. O ato de executar as atividades para alguma necessidade do usuário é o projeto de software. E todas as saídas que são produzidas enquanto as atividades estão sendo executadas são os produtos (um dos quais é o produto final).

Jalote (1997) descreve o processo de software como um tipo de abstração, e cada projeto é feito usando o processo como uma instância deste tipo. Em outras palavras, podem existir muitos projetos para um processo (ou seja, muitos projetos podem ser feitos usando um processo), e podem existir muitos produtos gerados a partir do projeto. Este relacionamento pode ser mais bem observado na Figura 4.

Figura 4: Processos, Projetos e Produtos



Fonte: Jalote (1997)

2.2 Modelos de Processo de Desenvolvimento de Software

No processo de desenvolvimento de software o foco das atividades está diretamente relacionada com a produção do software, como por exemplo, projeto, codificação e testes. Um modelo de processo de desenvolvimento especifica algumas atividades que devem ser executadas, bem como, especifica a ordem na qual elas devem ser efetuadas. (Jalote, 1997)

Como o processo de desenvolvimento especifica as principais atividades de desenvolvimento e de garantia de qualidade que precisam ser executadas no projeto, o

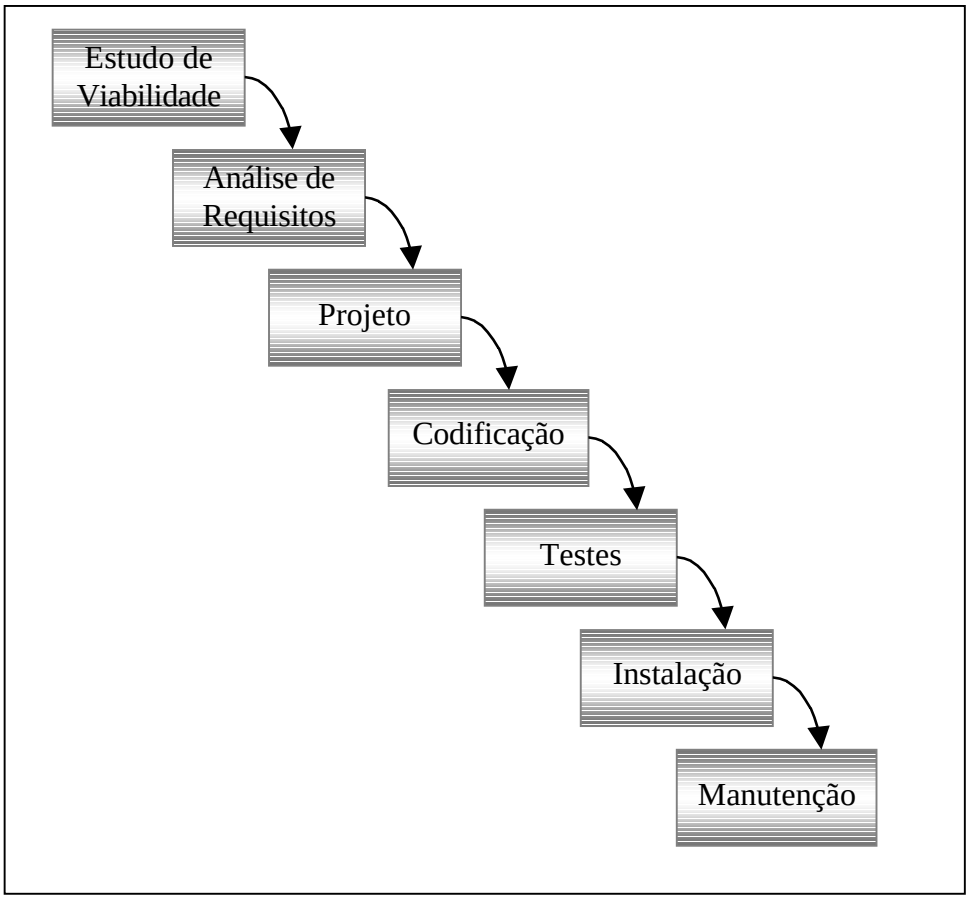
processo de desenvolvimento realmente forma o núcleo do processo de software. O processo de gerenciamento, abordado no capítulo 3, é decidido baseado no processo de desenvolvimento.(Jalote, 1997)

Devido à importância do processo de desenvolvimento, vários modelos têm sido propostos. As seções seguintes discutem alguns dos principais modelos.

2.2.1 Modelo Cascata

Segundo Ghezzi et al.(1991), o modelo cascata foi popularizado em 1970 e é abordado em diversos livros. Dentre os autores pesquisados neste trabalho, encontra-se o modelo cascata descrito por Ghezzi et al. (1991), Jalote (1997), Pressman (1995), Von Mayrhauser (1990) e Sommerville (1992).

Figura 5: Modelo Cascata



O modelo, que também pode ser denominado ciclo de vida clássico, é apresentado com algumas variações pelos diferentes autores. No entanto, apesar das variações o princípio é o mesmo e o modelo visualizado na Figura 5 pode ser

considerado um exemplo representativo, bem como as explicações posteriores se aplicam a todas variações.

2.2.1.1 Fases

A seguir é apresentada uma breve explicação das atividades realizadas em cada uma das fases do modelo cascata.

Fase de Estudo de Viabilidade

Segundo Ghezzi et al. (1991), o propósito desta fase é produzir um documento com o estudo de viabilidade que avalia os custos e benefícios da aplicação proposta. Para isto, primeiramente é necessário analisar o problema, pelo menos num nível global. Obviamente, quanto mais o problema for compreendido, melhor se poderá identificar as alternativas para solução, seus custos, e os seus benefícios para o usuário. Portanto, idealmente, se deveria executar uma análise do problema tão detalhadamente quanto necessário para o estudo de viabilidade. No entanto, o autor destaca que isto freqüentemente não ocorre na prática. O estudo é geralmente efetuado com um tempo limite e sobre pressão.

Em suma, o estudo de viabilidade deve antecipar o cenário para o desenvolvimento do software, resultando num documento que deve conter, conforme Ghezzi et al. (1991), no mínimo , os seguintes itens:

- Uma definição do problema.
- Alternativas de solução e os benefícios esperados.
- Recursos requeridos, custos e data para entrega de cada alternativa de solução proposta.

Fase de Análise de Requisitos

Esta fase deve identificar os atributos requeridos na aplicação, em termos de funcionalidade, performance, facilidade de uso, portabilidade, dentre outros. A pessoa responsável por esta fase deve relatar quais atributos a aplicação deve ter e não como obtê-los (isto é obtido através do projeto e codificação). Por exemplo, deve-se definir quais funções o software terá, sem utilizar estruturas de módulos ou algoritmos. (Ghezzi et al., 1991)

O documento com a especificação dos requisitos possui 2 propósitos: de um lado, deve ser analisado e confirmado com o usuário (cliente) para verificar se todas as expectativas foram abordadas e, de outro lado, é usado para desenvolver a solução que vá de encontro com os requisitos especificados.

O documento, conforme Ghezzi et al. (1991), pode ser produzido contendo os seguintes itens:

- **Requisitos funcionais**: descrevem o que o produto faz usando uma notação informal, semiformal, formal ou uma mistura delas. Vários livros, como por exemplo Ghezzi et al. (1991), Von Mayrhauser (1990), Jalote (1997) e Sommerville (1992), apresentam as notações existentes.
- **Requisitos não funcionais**: estes podem ser classificados dentro das seguintes categorias: credibilidade (segurança, integridade, dentre outros), precisão dos resultados, performance, questões de uso da interface homem-máquina, portabilidades, dentre outras.
- **Requisitos do processo de desenvolvimento e manutenção**: inclui propriedades de controle de procedimentos em particular, procedimentos para teste (priorizando os requisitos funcionais), procedimentos para manutenção, dentre outros.

Fase de Projeto

Segundo Von Mayrhauser (1990), enquanto a análise de requisitos se detém em o que o software a ser implementado fará, a fase de projeto descreve como a solução será implementada. O resultado, conforme Ghezzi et al. (1991), é a arquitetura do software: a função de cada módulo e o relacionamento entre os módulos.

Ghezzi et al. (1991) expõe que o exato formato do documento que especifica o projeto geralmente é padronizado pela compania.

Fase de Codificação

Conforme Von Mayrhauser (1990), o objetivo da fase de codificação é traduzir a solução em código. A autora indica ainda que a conclusão desta fase somente ocorre quando todo o código está escrito e documentado, compilado livre de erros e seguindo o

padrão do projeto. Além disto, até o final desta fase um plano de testes (descrevendo quando e como testar cada parte do código) deve ser traçado.

Fase de Testes

O código gerado deve ser testado rigorosamente baseado nos requisitos analisados, este é, segundo Von Mayrhauser (1990), o objetivo desta fase.

Von Mayrhauser (1990) apresenta os vários passos de testes a serem feitos. Primeiramente os módulos são testados isoladamente, denominado **teste de unidade**. Posteriormente, os módulos são testados em grupo visando verificar se estão interagindo adequadamente, denominado **teste de integração**. A partir destes testes executa-se o **teste de sistema** onde o software é testado em ambientes diferentes. O **teste de instalação** deve testar o software em sistemas com diferentes configurações. Além destes, ainda é requerido um **teste de aceitação**, que deve verificar juntamente com o usuário se os requisitos foram alcançados.

Fase de Implantação

A implantação, ou distribuição, geralmente ocorre em dois estágios. Num primeiro momento, a aplicação é distribuída para um grupo selecionado de usuários, para verificar o *feedback* dos usuários. Este tipo de teste é denominado beta teste. No segundo estágio, o produto é distribuído para todos os usuários.(Ghezzi et al., 1991)

Fase de Manutenção

A fase de manutenção, que se inicia a partir da entrega do software, é caracterizada pela realização de alterações de naturezas as mais diversas, seja para corrigir erros residuais da fase anterior, para incluir novas funções exigidas pelo cliente, ou para adaptar o software a novas configurações de hardware.

Sendo assim, Pressman (1995), caracteriza esta fase pelas seguintes atividades:

- **Correção** ou **Manutenção Corretiva**, a qual consiste da atividade de correção de erros observados durante a operação do sistema;
- **Adaptação** ou **Manutenção Adaptativa**, a qual realiza alterações no software para que ele possa ser executado sobre um novo ambiente

(CPU, arquitetura, novos dispositivos de hardware, novo sistema operacional, dentre outros);

- **Melhoramento Funcional** ou **Manutenção Perfectiva**, onde são realizadas alterações para melhorar alguns aspectos do software, como por exemplo, o seu desempenho, a sua interface, a introdução de novas funções, .
- **Manutenção preventiva** que ocorre quando o software é modificado para melhorar a confiabilidade ou a manutenibilidade futura, ou para oferecer uma base melhor para futuras ampliações. Esta atividade é caracterizada pelas técnicas de engenharia reversa e reengenharia.

A manutenção do software envolve, normalmente, etapas de análise do sistema existente (entendimento do código e dos documentos associados), teste das mudanças, teste das partes já existentes, o que a torna uma etapa complexa e de alto custo.

2.2.1.2 Análise Crítica do Modelo Cascata

Apesar de ser um modelo bastante popular, pode-se apontar algumas limitações apresentadas por este modelo, e citadas em Pressman (1995), Ghezzi et al. (1991), Jalote (1997) e Von Mayrhauser (1990):

1. o modelo assume que os requisitos são inalterados ao longo do desenvolvimento; isto em boa parte dos casos não é uma condição verdadeira, uma vez que nem todos os requisitos são completamente definidos na etapa de análise;
2. muitas vezes, a definição dos requisitos pode conduzir à definição do hardware sobre o qual o sistema vai funcionar; dado que muitos projetos podem levar diversos anos para serem concluídos, estabelecer os requisitos em termos de hardware é um tanto temeroso, dadas as freqüentes evoluções no hardware;
3. o modelo impõe que todos os requisitos sejam completamente especificados antes do prosseguimento das etapas seguintes; em alguns projetos, é às vezes mais interessante poder especificar completamente somente parte do sistema, prosseguir com o desenvolvimento do sistema, e só então encaminhar os requisitos de outras partes; isto não é previsto ao nível do modelo;

4. as primeiras versões operacionais do software são obtidas nas etapas mais tardias do processo, o que na maioria das vezes inquieta o cliente, uma vez que ele quer ter acesso rápido ao seu produto.

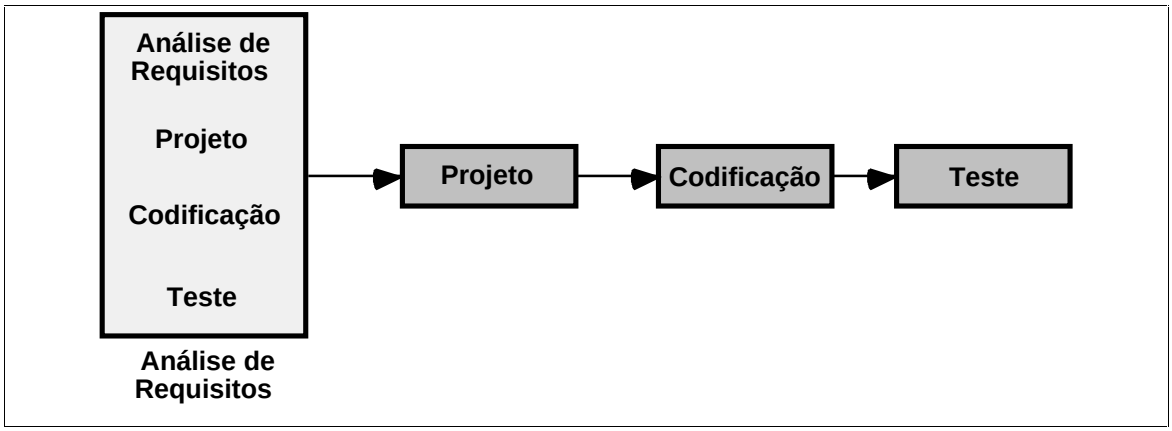
Em função destas limitações alguns autores apresentam o modelo cascata possuindo *feedback* entre as fases. Esta visão é abordada por Ghezzi et al. (1991) e Pressman (1995).

2.2.2 Prototipação

O objetivo do processo de desenvolvimento baseado na prototipação vai de contra as duas primeiras limitações do modelo cascata. A idéia básica da prototipação, de acordo com Jalote (1997), é que ao invés de manter inalterado os requisitos durante o projeto e codificação, um protótipo é desenvolvido para ajudar no entendimento dos requisitos. O desenvolvimento do protótipo, obviamente, passa por um projeto, codificação e teste, mas cada uma destas fases não é executada formalmente. E, com a utilização de um protótipo, o cliente é capacitado a entender melhor os requisitos desejados do sistema. Desta forma, resulta em requisitos mais estáveis (que serão alterados em uma freqüência menor).

A seqüência de eventos no processo de prototipação pode ser observado na Figura 6.

Figura 6: Modelo de Prototipação



Fonte: Jalote (1997)

O desenvolvimento do protótipo tipicamente inicia com uma versão preliminar do documento de especificação dos requisitos a ser desenvolvidos. Depois do protótipo desenvolvido, o usuário final e clientes têm a oportunidade de usarem o protótipo.

Baseado na experiência deles, é fornecido *feedback* para os desenvolvedores, informando o que está correto, o que precisa ser modificado, o que está faltando, o que não é necessário, dentre outros. Baseado no *feedback*, o protótipo é modificado incorporando as sugestões de mudança que podem ser feitas facilmente, e então os usuários e clientes utilizam novamente o protótipo. Este ciclo é repetido enquanto analistas e projetistas julgarem que o custo em se efetuar as alterações é válido para a elucidação dos requisitos. Ao final do ciclo, os requisitos iniciais são modificados para produzir a especificação final dos requisitos, os quais são utilizados para a produção do sistema.

2.2.2.1 Análise Crítica do Modelo de Prototipação

Pressman (1995) aponta alguns problemas na utilização do modelo baseado em prototipação:

- O cliente vê aquilo que parece ser uma versão do trabalho do software, desconhecendo que o protótipo se mantém unido "*com goma de mascar*", sem saber que, na pressa de colocá-lo em funcionamento, não levamos em consideração a qualidade global do software e a manutenibilidade a longo prazo. Quando informamos que o produto precisa ser reconstruído, o cliente exige que "alguns acertos" sejam aplicados para tornar o protótipo um produto de trabalho.
- O desenvolvedor muitas vezes faz concessões de implementação a fim de colocar um protótipo em funcionamento rapidamente. Concessões essas, que podem se tornar parte integrante do sistema.

No entanto, Pressman (1995) e Jalote (1997) destacam alguns benefícios da utilização deste modelo:

- O modelo é interessante para alguns sistemas de grande porte os quais representem um certo grau de dificuldade para exprimir rigorosamente os requisitos.
- Através da construção de um protótipo do sistema é possível demonstrar a realizabilidade do mesmo.
- É possível obter uma versão, mesmo simplificada, do que será o sistema com um pequeno investimento inicial.

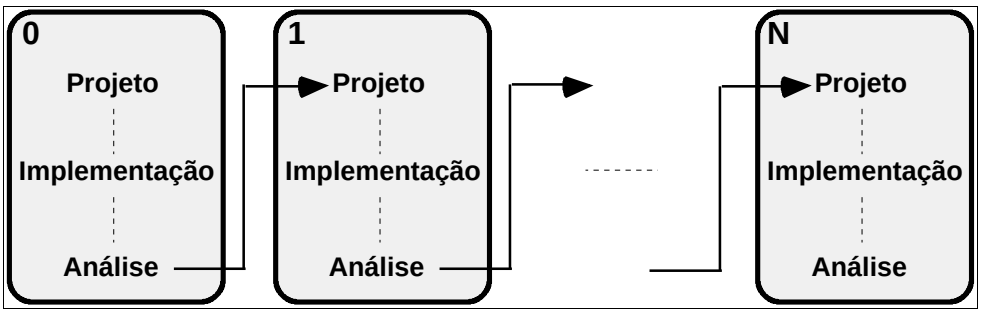
- A experiência de desenvolver o protótipo pode reduzir o custo das fases posteriores.

2.2.3 Modelo de Desenvolvimento Iterativo

O modelo iterativo vai de contra a terceira limitação do modelo cascata e tenta combinar os benefícios de ambos os modelos - prototipação e cascata. Segundo Jalote (1997), a idéia básica é que o software deve ser desenvolvido incrementalmente, onde a cada incremento alguma funcionalidade é adicionada ao sistema até que o sistema completo esteja implementado. A cada passo pode-se executar modificações no projeto.

Ghezzi et al. (1991) ressalta que este modelo pode ser denominado também de evolucionário, incremental ou *delivery*. A Figura 7 ilustra o modelo.

Figura 7: Modelo Iterativo



Fonte: Jalote (1997)

No primeiro passo do modelo, uma implementação inicial é feita para um subconjunto do problema. Este subconjunto contém alguns aspectos chaves do problema que é fácil de entender e implementar e cuja forma seja útil e usável. Uma **lista de controle de projeto** é criada. Esta lista contém, em ordem, todas as tarefas que devem ser executadas para obter a implementação final. Através da lista de controle de projeto é possível saber quão longe se está da conclusão do sistema.

Cada passo consiste de remover a próxima tarefa da lista, projetar e implementar a tarefa selecionada, codificar e testar a implementação, efetuar a análise do sistema parcial obtido após este passo, e atualizar a lista com o resultado da análise. Estas três fases são denominadas fase de projeto, fase de implementação e fase de análise. O processo é iterativo até que a lista de controle de projeto esteja vazia.

2.2.3.1 Análise Crítica do Modelo Iterativo

Jalote (1997) destaca que uma vantagem desta abordagem é a facilidade em testar o sistema, uma vez que a realização de testes em cada nível de desenvolvimento é, sem dúvida, mais fácil do que testar o sistema final. Além disso, como na prototipação, a obtenção de um sistema, mesmo incompleto num dado nível, pode oferecer ao cliente interessantes informações que sirvam de subsídio para a melhor definição dos requisitos finais do sistema.

No entanto, Jalote (1997) explica que, um problema prático com este tipo de desenvolvimento encontra-se na elaboração do contrato – como o custo das características adicionais será determinada e negociada.

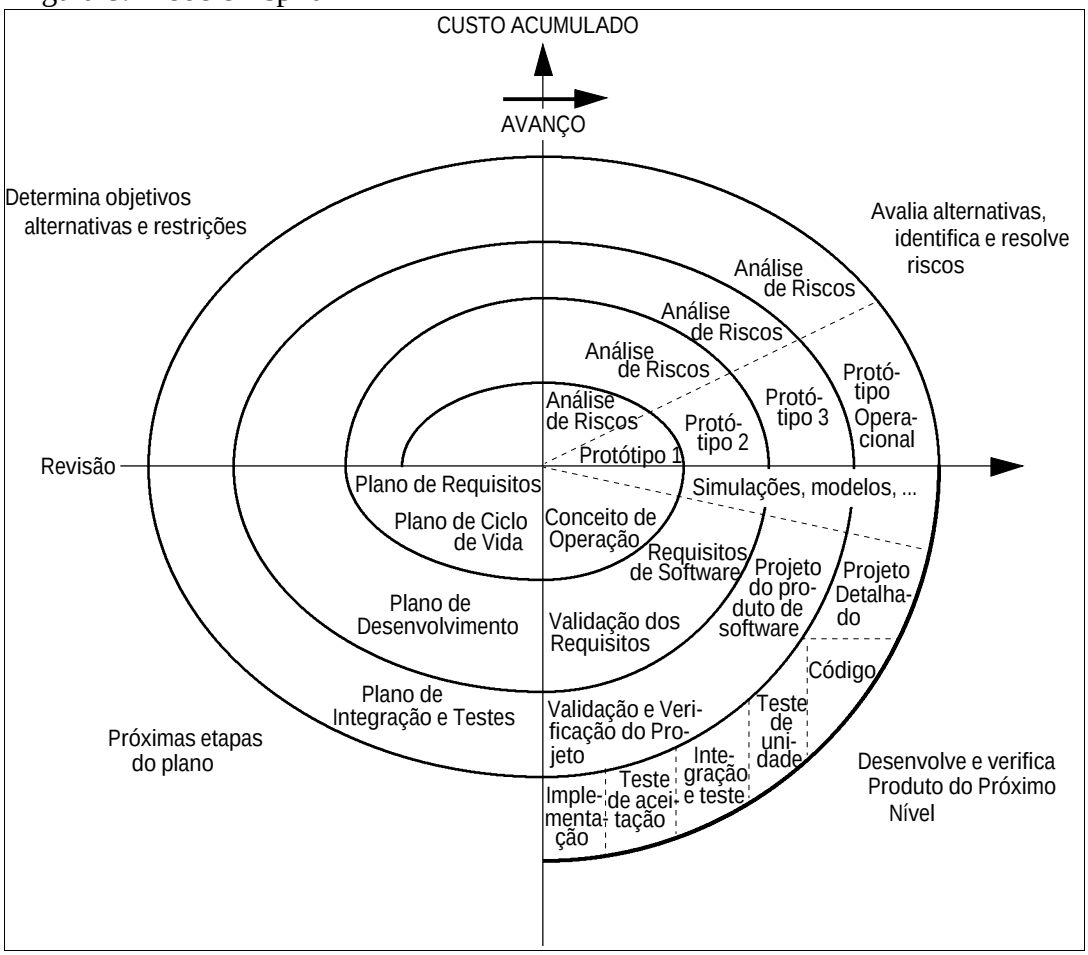
2.2.4 Modelo Espiral

"O objetivo do modelo espiral para o processo de produção de software é prover um *framework* para projetar processos, guiado pelo nível de risco do projeto... o modelo espiral pode ser visto como um metamodelo, porque ele pode acomodar qualquer modelo de processo de desenvolvimento" (Ghezzi et al., 1991). A Figura 8 ilustra o modelo espiral, que é descrito na seqüência.

Cada ciclo do espiral inicia com identificação dos objetivos para aquele ciclo, e as diferentes alternativas possíveis para alcançar os objetivos bem como, as restrições que existem. Este é o primeiro quadrante do ciclo (superior esquerdo). O próximo passo no ciclo é avaliar as diferentes alternativas baseado nos objetivos e restrições. O foco da avaliação neste passo é baseado na percepção do risco para o projeto (definições de risco são apresentadas na seqüência). O próximo passo é desenvolver estratégias que elimine as incertezas e os riscos. Este passo envolve atividades como *benchmark*, simulação e prototipação. O próximo passo é desenvolver, mantendo em mente os riscos. Finalmente o próximo estágio é planejado.

Para Jalote (1997), risco reflete as chances de algum dos objetivos do projeto não ser alcançado. Já para Ghezzi et al. (1991) riscos são circunstâncias potencialmente desfavoráveis que podem prejudicar o processo de desenvolvimento e a qualidade do produto. Questões relacionadas com o gerenciamento do risco são apresentadas no capítulo 3.

Figura 8: Modelo Espiral



Fonte: Jalote (1997)

2.2.4.1 Análise Crítica do Modelo Espiral

Segundo Pressman (1995), as vantagens deste modelo são:

- O paradigma de modelo espiral para a Engenharia de Software (ES) atualmente é a abordagem mais realística para o desenvolvimento de sistemas e de software em grande escala.
- Usa uma abordagem “evolucionária” à engenharia de software, capacitando o desenvolvedor e o cliente a entender e reagir aos riscos em cada etapa evolutiva.
- O modelo se adequa a sistemas que representem um alto risco de investimento para o cliente.

No entanto, Pressman (1995) observa que alguns cuidados devem ser tomados:

- Pode ser difícil convencer grandes clientes de que a abordagem evolutiva é controlável.
- Exige considerável experiência na avaliação dos riscos e fia-se nessa experiência para o sucesso.
- O modelo é relativamente novo (1988) e demorará algum tempo até que a eficácia desse novo paradigma possa ser determinada com certeza absoluta (vale relatar que não foi encontrado na literatura textos posteriores apresentando a eficácia do modelo).

2.3 Considerações Finais

É importante notar que, conforme exposto por Sommerville (1996), não existem coisas certas ou erradas no processo de software. Diferentes processos de software organizam as atividades de diferentes maneiras. Diferentes organizações usam diferentes processos para produzir o mesmo tipo de produto. E, diferentes tipos de produtos podem ser produzidos por uma organização utilizando o mesmo tipo de processo.

Desta forma, os modelos apresentados servem como guia para as organizações desenvolverem o seu processo de desenvolvimento.

Finalizando, vale ressaltar que o processo de software compreende o processo de desenvolvimento de software, ênfase deste capítulo, e o processo de gerenciamento de projeto que é abordado no capítulo seguinte e se constitui o principal foco deste trabalho.

3 GERENCIAMENTO DE PROJETOS

O gerenciamento de projetos de software é uma tarefa de fundamental importância no processo de desenvolvimento de um produto, sendo definido como uma primeira camada deste processo. O gerenciamento de projeto não é visto como uma etapa clássica do processo de desenvolvimento uma vez que ele acompanha a todas as etapas, da concepção à obtenção do produto. (Pressman, 1995)

Project Management Institute (*apud* O'Connor 2000) define gerenciamento de projeto como "a aplicação de conhecimentos, habilidades, ferramentas e técnicas para projetar atividades a fim de satisfazer ou superar as expectativas dos *stakeholders* em relação a um projeto".

No entanto, são vários os problemas relacionados ao desenvolvimento de software, os quais são resultantes da omissão ou do mau uso de metodologias e técnicas adequadas a esta importante tarefa de engenharia. Chang & Christensen (1999) citam que são muitos os problemas dos gerentes de projetos de software para a realização de grandes sistemas, incluindo: (i) reunir, treinar e motivar uma grande equipe; (ii) desenvolver ou adotar processos de gerenciamento e engenharia; (iii) desenvolver e manter requisitos; (iv) planejar, orçar e monitorar o projeto; (v) identificar e resolver conflitos de recursos e (vi) monitorar o projeto inteiro continuamente.

Royce (1998) relata o resultado de três importantes análises, realizada na metade dos anos 90, sobre o estado da engenharia de software nas empresas. As três análises chegaram a mesma conclusão, resumida em:

- Desenvolvimento de software ainda é altamente imprevisível. Somente aproximadamente 10% dos projetos de software são entregues com sucesso (considerando o orçamento e cronograma estimado).
- Disciplina de gerenciamento é mais um discriminador do sucesso ou fracasso que uma tecnologia avançada.
- O nível de re-trabalho é um indicativo de um processo imaturo.

Pressman (1995) afirma que boa parte dos fracassos no que diz respeito aos projetos de software deve-se, principalmente, a problemas de administração ou gerenciamento do desenvolvimento de software.

Sommerville (1992) e O'Connor (2000) atribuem a dificuldade de gerenciamento de projetos de software à diferença existente em relação a outros tipos de projetos, podendo-se destacar 5 aspectos:

- Software não é "palpável"/visível: quando o produto a ser desenvolvido é uma estrada, por exemplo, o progresso pode ser claramente verificado. No software o progresso não é visível, o gerente de projeto depende da documentação para verificar o progresso do projeto.
- Não se tem o entendimento claro do processo de software: na engenharia de software os modelos de desenvolvimento de software (como os apresentados na seção 2.2) são representações simplificadas do processo.
- Vários projetos de software são "projetos únicos", ou seja, não há semelhança com projetos previamente desenvolvidos. Nestes casos, a experiência histórica é um valor limitante em predizer como o projeto deverá ser gerenciado.
- Complexidade: "Cada dólar, peso ou euro gasto em um produto de software contem mais complexidade que outros artefatos da engenharia." (O'Connor 2000).
- Flexibilidade: a facilidade com a qual um software pode ser alterado é geralmente visto como uma de suas vantagens.

Na seqüência são apresentados os parâmetros e atividades que caracterizam gerenciamento de projeto de software.

3.1 Alguns Dados Históricos

Uma significante reengenharia do processo de desenvolvimento de software ocorreu nos últimos 20 anos, motivada pelo aumento da demanda de softwares produzidos mais rapidamente e custos reduzidos. Desta forma, muito do gerenciamento convencional e técnicas até então utilizadas tem sido substituídas por novas abordagens

que combinam experiências de sucesso em gerenciamento de projetos com os avanços da engenharia de software.

Para Royce (1998) um gerenciamento moderno de software está baseado em diversos princípios, podendo-se priorizar: (i) processo baseado em uma abordagem em que primeiramente deve-se definir a arquitetura; (ii) estabelecer um processo de desenvolvimento iterativo; (iii) métodos de projeto devem enfatizar o desenvolvimento baseado em componentes; (iv) estabelecer um ambiente de gerenciamento flexível e (v) utilização de ferramentas automatizadas que suportam diferentes formatos.

Ainda no que se refere a uma visão histórica do gerenciamento de projetos de software, um comparativo realizado por Jones (*apud* Royce, 1998) entre tecnologias que levaram ao sucesso ou insucesso 1000 (mil) projetos agrupados dentro de seis sub-indústrias: sistemas de software, sistemas de informação, software comercial, *outsource* software, software militar e software para usuários finais; reafirmam a importância do gerenciamento. Com base neste levantamento Jones (*apud* Royce, 1998) relata que "seis dos dezessete fatores tecnológicos associados com desastres no projeto de software são falhas específicas no domínio do gerenciamento de projetos, e três das outras tecnologias deficientes podem ser indiretamente determinadas por práticas pobres de gerenciamento". O Quadro 2 retrata os fatores relacionados com o gerenciamento de projetos.

Quadro 2: Tecnologias Usadas em Projetos de Software

Tecnologia em projetos mal sucedidos	**Tecnologia em projetos bem sucedidos**
Ausência de dados históricos de medição dos softwares	Precisa medição do software
Insucesso ao usar ferramentas automatizadas de estimativa	Ferramentas de estimativa usadas facilmente
Fracasso no uso de ferramentas automatizadas de planejamento	Uso contínuo de ferramentas de planejamento
Falhas na monitoração do progresso do projeto ou *milestones*	Relatório formal do progresso do projeto
Falha no uso de uma arquitetura efetiva	Planejamento formal da arquitetura
Não cumprimento de um método de desenvolvimento	Método formal de desenvolvimento
Gerenciamento de risco informal	Gerenciamento de risco formal
Insucesso no uso formal do controle de configuração	Controle de configuração automatizado

Mais de 30% abaixo dos requisitos do usuário	Menos de 10% abaixo dos requisitos do usuário

Fonte: Royce (1998)

Um relatório apresentado pelo Standish Group (*apud* Royce, 1998) focado na indústria de software comercial concluiu que:

- Companhias americanas gastariam $81 bilhões em projetos de softwares cancelados em 1995.
- 31% dos projetos de software estudados foram cancelados antes deles serem completados.
- 53% dos projetos de software passam do limite em mais de 50%.
- Somente 9% dos projetos de software das companhias grandes foram distribuídos no prazo e dentro do orçamento. Para médias e pequenas empresas, os números aumentaram para 16% e 28%, respectivamente.

Assim como relatado anteriormente por Jones (*apud* Royce, 1998), este relatório cita que a principal razão para o sucesso ou fracasso está centrada nos requisitos do processo de gerenciamento. Desta forma, fica evidenciada e ressaltada a importância do gerenciamento de software para o sucesso dos projetos.

3.2 Parâmetros/Atividades

Pressman (1995) e Jalote (1997) definem que para um projeto de software ser bem sucedido, é necessário que alguns parâmetros sejam bem analisados, como por exemplo, o escopo do software, os riscos envolvidos, os recursos necessários, as tarefas a serem realizadas, os marcos de referência a serem acompanhados, os esforços (custos) aplicados e a sistemática a ser seguida. A análise de todos estes parâmetros é a função típica do gerenciamento de projetos, função esta que se inicia antes do trabalho técnico e que prossegue à medida que o software vai se concretizando na forma de um produto.

Sommerville (1992) acrescenta a estes parâmetros algumas atividades que também são de responsabilidade do gerente de projeto, que são a seleção e avaliação da equipe e elaboração de relatórios. Já Jalote (1997) cita a determinação do *schedule* e *milestone* como atividades pertinentes ao gerenciamento.

3.2.1 Fase Inicial do Gerenciamento

Nesta primeira fase do gerenciamento, o objetivo é levantar alguns pontos importantes para a boa condução do projeto: objetivos e escopo do software, alternativas de solução, restrições administrativas e técnicas.

O conhecimento destas informações vai permitir ter uma primeira estimativa do projeto em termos de custo, além de proporcionar uma melhor divisão no que diz respeito às tarefas do projeto, estabelecendo, inclusive, subsídios para que se possa avaliar o estado de evolução do projeto. (Pressman, 1995)

Nesta fase, as duas partes envolvidas no projeto (o desenvolvedor e o cliente) devem reunir-se para definir os objetivos e o escopo do projeto

É importante destacar que o levantamento dos objetivos do projeto não levarão em conta, nesta fase, como eles serão alcançados. A definição do escopo do software permitirá obter, num nível elevado de abstração, as principais funções a serem supridas pelo produto. (Jalote, 1997)

Somente após estes dois pontos terem sido levantados, é que serão discutidas as alternativas de solução, como forma de melhor selecionar a abordagem a ser adotada de modo que permita respeitar as restrições impostas: prazo de entrega, orçamento, disponibilidade de pessoal, dentre outros.

3.2.2 Medições e Métricas

Quando se considera boa parte dos empreendimentos técnicos, verifica-se que as medições e as métricas permitem um melhor entendimento do processo utilizado para desenvolver um produto, assim como uma melhor avaliação do próprio produto, consenso encontrado em (Candéas & Lopes, 1999), (IFPUG, 2000), (Software Productivity Research, 2000) e (Pressman, 1995).

A quantificação dos aspectos relacionados ao processo de obtenção de um produto, assim como do produto, é importante, pelas seguintes razões:

- No caso do processo de desenvolvimento, as medições podem permitir melhorias no processo, aumentando a sua produtividade;

- No caso do produto, as medições podem proporcionar informações a respeito de sua qualidade.

Apesar de importante, nem sempre a medição é uma tarefa evidente, conduzindo a diversos questionamentos:

- Quais são as métricas mais adequadas para o processo e para o produto?
- Como os dados obtidos serão processados?
- É justo utilizar medições para se comparar pessoas, processos e produtos?

3.2.3 Estimativa

Conforme Candéas & Lopes (1999), a estimativa é um exercício importante, especialmente para o planejamento do projeto de software. Fatores como o esforço humano exigido (pessoas/mês), duração cronológica do projeto, custo, e outros devem ser levantados neste momento. O problema é como definir estes fatores.

Em grande parte dos casos, as estimativas são feitas com base na experiência passada. No caso de se ter um projeto relativamente similar a um projeto já realizado, não fica difícil estimar questões como esforço, cronograma e custo, uma vez que estes serão muito próximos daqueles relativos ao projeto anterior. (Pressman, 1995)

Por outro lado, a estimativa através desta abordagem pode tornar-se complexa se o novo projeto apresenta características inovadoras com relação ao(s) projeto(s) anterior(es).

Apesar da existência de diversas técnicas de estimativa, é importante destacar algumas de suas características comuns, conforme encontrado em Software Productivity Research (2000):

- o escopo do projeto é estabelecido previamente;
- são utilizadas métricas de software e histórico de aferições passadas como base das estimativas;
- o projeto é dividido em pequenas partes as quais são estimadas individualmente.

Em alguns casos, os gerentes de projeto utilizam mais de uma técnica de estimativa, de modo que os resultados obtidos são comparados para verificar a coerência dos cálculos realizados.

3.2.4 Análise de Riscos

Em qualquer projeto de engenharia, e os projetos de software não fogem a esta regra, existe um conjunto de incertezas, citadas em (Pressman, 1995):

- a necessidade de um computador é algo realmente claro?
- as funções a serem implementadas estarão prontas antes do prazo final?
- que problemas técnicos serão enfrentados ao longo do projeto?
- eles serão contornados de modo a cumprir o cronograma?
- as mudanças de requisitos que eventualmente aparecerão não provocarão um aumento considerável no tempo (e nos custos) de desenvolvimento?

A análise de riscos é uma tarefa de grande importância no gerenciamento de um projeto de software, embora, em muitos casos, esta atividade nem seja considerada.

O seu objetivo é determinar um conjunto de passos a serem seguidos para determinar os riscos envolvidos no projeto: identificação, avaliação, classificação, definição de estratégias para administrar os riscos, resolução dos riscos, dentre outros.

3.2.5 Determinação de Prazos

Outro aspecto importante relacionado ao gerenciamento de um projeto de software, ressaltado por Pressman (1995), é a definição dos prazos de desenvolvimento. Cada projeto de software é caracterizado por um conjunto de atividades a serem executadas.

Como qualquer outro projeto de engenharia, o projeto de um software deve obedecer a um conjunto bem definido de tarefas, ao inter-relacionamento das tarefas, sendo que um determinado esforço será dedicado à realização de cada tarefa envolvendo a alocação de um determinado grupo de pessoas e outros recursos.

Com base nestas informações, é possível criar uma representação (gráfico ou tabela) que exprima o tempo dedicado a cada tarefa, determinando assim o prazo final do projeto. As seções 3.3.2.1 e 3.3.2.2 apresentam as técnicas (representações) para auxiliar nesta atividade.

3.2.6 Milestones

Durante o planejamento de um projeto, Sommerville (1992) comenta que, uma série de *milestones* (marcos de referência) deve ser estabelecida. *Milestone* é um ponto final de uma atividade do processo de desenvolvimento de software, portanto, os *milestones* devem representar o final de um estágio distinto do projeto. Sendo que para cada *milestone* um relatório deve ser elaborado.

Um bom *milestone* é caracterizado pela elaboração da documentação, por exemplo, 'plano de testes formulado'. *Milestones* não deve ser indefinido, por exemplo, um pobre *milestone* mal definido seria '80% do código concluído'. Pois não há maneira objetiva de se afirmar que 80% do código foi concluída.

Milestones não precisam ser estabelecidos para cada atividade do projeto. Uma regra prática, citada por Sommerville (1992), é que os *miliestones* devem estar agendados em intervalos de 2-3 semanas, podendo variar dependendo do processo de desenvolvimento seguido.

3.2.7 Scheduling

Desenvolver o cronograma de um projeto é uma das tarefas mais difíceis do gerenciamento de software, afirma Sommerville (1992; 1996), pois envolve estimar o tempo e recursos necessários para completar as atividades/tarefas e organiza-las em uma seqüência coerente. A menos que o *scheduling* do projeto seja similar a um projeto prévio, estimativas prévias não são uma boa base. Diferentes projetos usam diferentes linguagens de programação e metodologias, o qual complica a tarefa de estimar o *schedule.*

O cronograma de projetos envolve separar o trabalho envolvido no projeto em tarefas separadas e avaliar quando estas tarefas serão concluídas. Geralmente algumas destas tarefas são realizadas em paralelo. Gerentes de projetos devem coordenar estas

tarefas paralelas e organizar o trabalho evitando situações onde o projeto inteiro é atrasado em decorrência a uma tarefa crítica não concluída.

A regra prática, citada em Sommerville (1992), para estimar o *schedule* é estimar como se nada dará errado, incrementar a estimativa para cobrir problemas previstos e então adicionar um "fator contingência" para cobrir problemas não previstos. Este "fator contingência" depende do tipo do projeto, qualidade e experiência da equipe.

Para estimar o tempo total requerido pelo projeto, Sommerville (1992) comenta que pode-se usar o tamanho estimado do sistema e dividir pela produtividade esperada da equipe. Técnicas usadas em estimativas são apresentadas no capítulo 4.

O resultado do processo de *scheduling* são gráficos relacionando as tarefas, dependências entre tarefas e alocação de pessoal. A seção 3.3.2.1 e 3.3.2.2 apresenta os gráficos utilizados: gráfico de PERT (*Program Evaluation and Review Technology*) e gráfico de Gantt.

3.2.8 A Monitoração e Controle

Uma vez definida a programação do desenvolvimento (*schedule*), a atividade de monitoração e controle do projeto será iniciada e permanecerá durante todo o projeto, segundo Pressman (1995). Cada tarefa prevista no programa é monitorada pelo gerente de projeto. Caso o desenvolvimento desta não esteja de acordo com a programação, o gerente pode atuar no sentido de avaliar o impacto do não cumprimento dos prazos para o projeto como um todo ou atuar sobre ela no sentido de fazer com que a sua realização aproxime-se do que foi programado. Isto pode ser feito pela alocação de mais recursos para o cumprimento da tarefa.

Ainda, em função desta avaliação, tarefas podem ser reorganizadas no sentido de que o projeto global, a despeito dos atrasos envolvidos em algumas tarefas, possa ser concluído dentro do prazo final estipulado.

3.2.9 Seleção e Avaliação da Equipe

O gerente de projeto geralmente tem a responsabilidade de selecionar a equipe que trabalhará no projeto. O ideal, segundo Sommerville (1992), é ter pessoal com

habilidade e experiência para realizar o projeto. No entanto, o autor assegura que na maioria dos casos isto é inalcançável devido as seguintes razões:

- O orçamento pode tornar impossível o uso de pessoal com salário alto.
- Pessoal com a experiência apropriada pode simplesmente não estar disponível.
- A organização pode exigir que pessoal em treinamento trabalhe no projeto para adquirir experiência.

O gerente de projeto tem que trabalhar com estas hipóteses quando selecionar a equipe.

3.2.10 Elaboração de Relatórios

Embora todos os membros da equipe tenham responsabilidade de gerar documentação, o gerente de projeto é o principal responsável por relatar o projeto ao cliente e ao superior na organização. Gerentes de projeto devem ser capazes de escrever documentos concisamente e coerentemente apresentando detalhadamente as características do projeto. Além de apresentar estas informações durante o processo de desenvolvimento. (Sommerville, 1992)

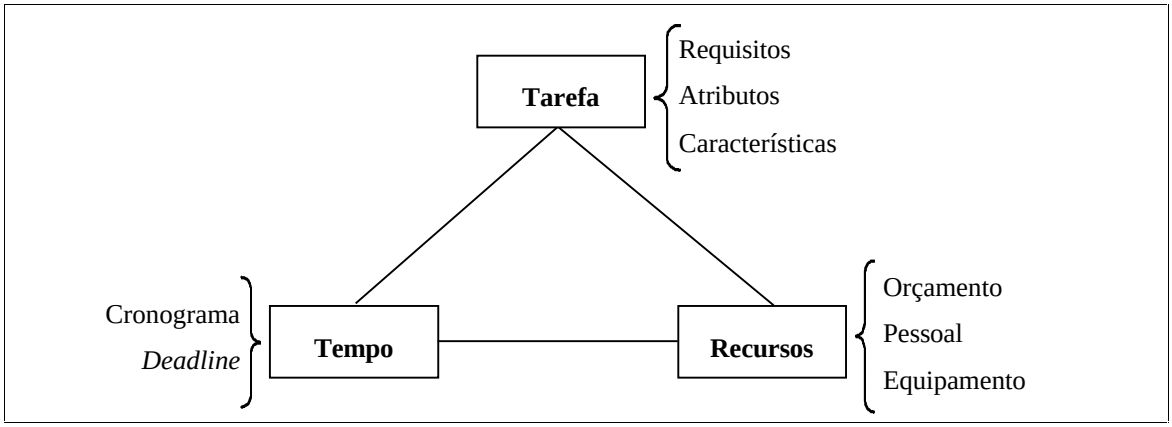
3.3 Dimensões do Gerenciamento de Projeto

Gerenciamento de projeto freqüentemente acarreta várias questões conflitantes e preocupações, tais como: não há tempo para executar a tarefa, o trabalho é muito complexo ou o orçamento não é adequado. Para proceder nestas situações, Strauss (1997) recomenda que, deve-se entender e considerar as 3 dimensões gerais do gerenciamento de projeto: tempo, tarefa e recursos. O autor destaca ainda que “sem um entendimento de como estes 3 fatores se inter-relacionam, o gerente pode facilmente entrar em modo reativo, constantemente respondendo a crise do momento.”

Estes 3 fatores constantemente interagem em um projeto, mudando a prioridade e variando em importância conforme o projeto avança. Entender como estes fatores interagem fornece uma perspectiva objetiva do processo de desenvolvimento. Portanto, esta é uma tarefa típica do gerente de projeto - gerenciar estes fatores e tomar as decisões.

Jim McCarthy (*apud* Strauss, 1997) da Microsoft reforça este conceito afirmando que "como um gerente de desenvolvimento, se está trabalhando com 3 coisas: recursos (pessoas e dinheiro), características (o produto e sua qualidade), e o cronograma. Este triângulo de elementos é tudo com o que se trabalha. Não existe nada mais para ser trabalhado." As três dimensões e sua interação pode ser visualizada na Figura 9.

Figura 9: Dimensões do Gerenciamento de Projeto



Fonte: Strauss (1997)

O 'Connor (2000) também define as responsabilidades do gerente de projetos utilizando estas 3 dimensões. Para o autor a responsabilidade do gerente é "planejar e programar o desenvolvimento de software. Ele deve supervisionar o trabalho garantindo a realização dos requisitos e monitorando o progresso verificando se o desenvolvimento está no tempo previsto e dentro do orçamento."

As seções seguintes definem cada uma das dimensões e exemplificam a forma de interação entre elas.

3.3.1 Tempo

O tempo requerido refere-se ao cronograma – especialmente ao *deadline* (data final). Esta data depende da natureza da tarefa (projeto) e da disponibilidade de recursos (pessoas e recursos).

Como regra, Strauss (1997) considera adequado dizer que, quanto mais recursos disponíveis e mais simples o projeto, mais rápido a tarefa pode ser completada. No entanto, o autor também ressalta que, adicionar mais pessoas não reduz o tempo na

mesma proporção, devido ao *overhead* de comunicação e administração para coordenar todas as atividades.

3.3.2 Tarefa

A tarefa se refere ao o que exatamente está sendo desenvolvido. É o escopo do trabalho a ser realizado: a grandeza e a complexidade da aplicação final. Ou seja, consiste na especificação dos requisitos, no projeto funcional, manuais do usuário, dentre outros.

A definição do produto final, segundo Strauss (1997), determinará o número de pessoas necessárias para produzir a aplicação, as habilidades das pessoas, o tipo de equipamento e quanto tempo levará para completar o projeto.

3.3.3 Recursos

Recursos basicamente se referem a quanto dinheiro está disponível para ser gasto no projeto e como o dinheiro é aplicado em termos de pessoas, material e equipamento.

Strauss (1997) cita que, em geral, quanto mais dinheiro disponível, mais rápido a aplicação pode ser desenvolvida e maior a qualidade ou mais elaborada ela pode ser. Entretanto, o autor ressalta que, se um projeto está atrasado no cronograma, adicionar recursos (pessoas e dinheiro) nem sempre aumenta a velocidade de desenvolvimento. Adicionando pessoas para um projeto atrasado pode causar um atraso maior ainda se os recursos errados são adicionados ou são adicionados no tempo errado.

3.4 Fases do Processo de Gerenciamento

Jalote (1997) enquadra as atividades, descritas na seção 3.1, dentro de três grandes fases do processo de gerenciamento: (i) planejamento; (ii) monitoramento e controle; (iii) análise conclusiva. No entanto, Ghezzi et al. (1991), ressalta a importância da organização e do gerenciamento do risco.

3.4.1 Planejamento

O objetivo desta seção é descrever sobre a importância do planejamento e do *schedule* para o gerenciamento de projetos. Pois, conforme citado por Jalote (1997) e

Sommerville (1992), “estas atividades freqüentemente absorvem a maioria dos esforços gerenciais de um projeto de software.” A importância do planejamento também é ressaltada por Wu & Simmons (2000) ao afirmar que “sem um planejamento realista e objetivo do projeto do software, o processo de desenvolvimento não pode ser gerenciado de maneira efetiva”.

Esta fase tem como objetivo, segundo Ghezzi et al. (1991), identificar todos os requisitos do projeto, para então, conforme Jalote (1997) e Ghezzi et al. (1991), desenvolver um planejamento para o desenvolvimento do software cujos objetivos do projeto possam ser alcançados eficientemente. Sendo desta forma, o planejamento do projeto considerado base fundamental para o gerenciamento.

O gerenciamento efetivo de projeto de software depende de um planejamento completo do progresso do projeto, sendo portanto, desenvolvido antes das atividades de desenvolvimento. Este planejamento, segundo Sommerville (1992), deve antecipar problemas que podem surgir e preparar soluções com antecedência. O plano redigido no início do projeto deve ser usado para dirigir o projeto. Este plano inicial não é estático, e sim, deve ser modificado conforme a evolução do projeto. (O’Connor, 2000)

Uma das questões a ser definida nesta fase é determinar qual o modelo de processo de desenvolvimento (discutido no capítulo 2) é adequado ao projeto. Outra decisão crítica é determinar os recursos (quantidade e habilidade das pessoas a serem envolvidas no processo e recursos computacionais) necessários para o projeto. Por conseguinte, o custo do projeto é diretamente proporcional ao número de pessoas envolvidas no projeto. (Ghezzi et al., 1991)

O problema de “predizer/prever” quantas pessoas e outros recursos são necessários para um dado projeto é conhecido com estimativa de custos de software, assunto este abordado no capítulo 4.

Jalote (1997) e Wu & Simmons (2000), acrescenta às questões abordadas por Guezzi, a determinação do *schedule* e *milestones* e a definição da equipe como atividades pertinentes a esta fase do gerenciamento.

Para Ghezzi et al. (1991), um dos requisitos básicos do gerenciamento é medir a produtividade das pessoas e processos envolvidos na produção. As medidas obtidas são usadas durante a fase de planejamento do projeto como base para estimar recursos.

Estimar a produtividade dos programadores é importante por 2 razões, ressalta Sommerville (1992):

- Sem uma estimativa de produtividade é impossível prever um cronograma confiável.
- Os benefícios de usar técnicas da engenharia de software e ferramentas podem demonstrar aos superiores que o gerente usa resultados para melhorar a produtividade em todo o ciclo de vida do software.

Devido ao fato do software não ser "palpável" não é possível medir a produtividade diretamente. Produtividade em sistemas manufaturados pode ser medida contando o número de unidades produzidas dividido pelo número de horas trabalhadas. Em sistemas computacionais, o que se espera é estimar o custo de um sistema dado a sua funcionalidade.

A técnica de pontos por função (detalhada na seção 4.3) é uma forma de medir o software através da sua funcionalidade. No entanto, Sommerville (1992) ressalta que "métricas de produtividade são somente um guia subjetivo e relativo de produtividade. Os gerentes de projeto devem acrescentar percepção e intuição para estimar a real produtividade."

3.4.2 Monitoramento e Controle

Segundo Jalote (1997), trata-se da fase mais longa do processo de gerenciamento, abrangendo a maior parte do processo de desenvolvimento. Para Ghezzi et al. (1991), o propósito desta fase é monitorar o progresso das atividades planejadas e garantir que os objetivos estão sendo executados.

Custo, *schedule* e tarefas são os principais fatores a serem gerenciados, portanto, o maior esforço desta fase envolve monitorar e controlar estes fatores. Outra importante atividade desta fase é monitorar os riscos potenciais do projeto. Se as informações obtidas pelo monitoramento sugerem que os objetivos do projeto podem não ser alcançados, faz parte das atividades desta fase planejar as ações para controlar as atividades de desenvolvimento. (Jalote, 1997)

O monitoramento e controle do processo requerem informações referentes ao projeto. Tais informações são obtidas pelo processo de gerenciamento a partir do processo de desenvolvimento, sendo a interpretação das informações parte do monitoramento e controle. (Jalote, 1997)

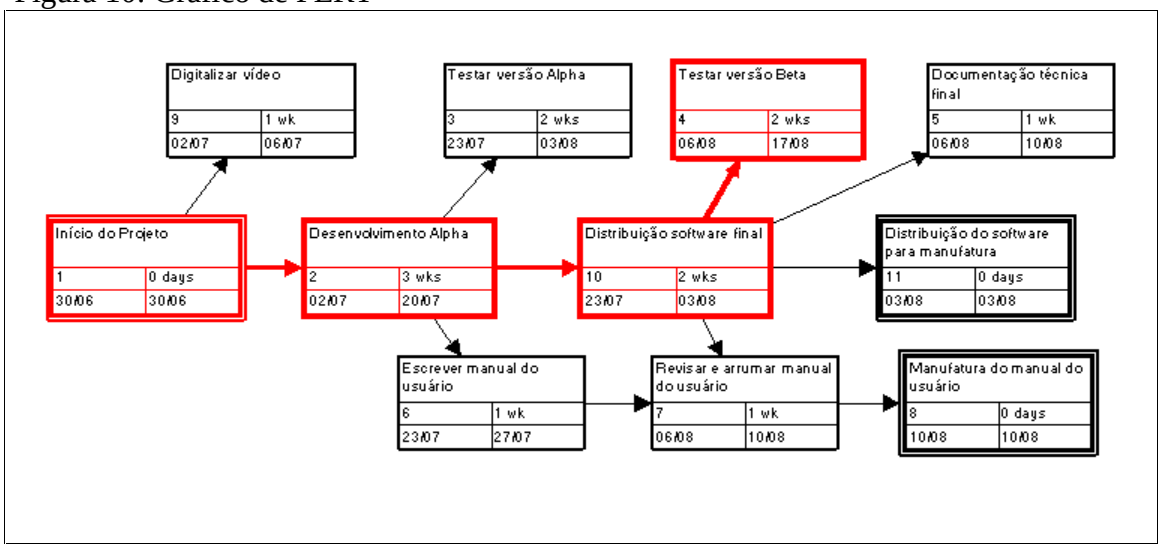
Knotts et al. (1998) observa que a ferramenta original para se monitorar um projeto é o CPM (*Critical Path Method*) e PERT. Segundo Ghezzi et al. (1991), o gráfico de PERT e o gráfico de Gantt são as 2 principais técnicas para monitoramento e controle no processo de gerenciamento de projetos, pois são úteis para monitorar o progresso do projeto em relação ao planejado e detectar os desvios a partir deles.

3.4.2.1 Gráfico de PERT

O gráfico de PERT é uma rede de caixas interconectadas, onde cada uma representa uma tarefa ou um *milestone* no processo de desenvolvimento. Através da identificação de quais tarefas são pré-requisitos para outra tarefa, pode-se verificar a interdependência entre as diferentes atividades do projeto. (Sommerville, 1992)

Este gráfico é útil para expor as dependências entre as tarefas e gargalos no fluxo de trabalho que de outro modo poderiam não ser identificados, ressalta Strauss (1997). Um exemplo do gráfico de PERT pode ser visualizado na Figura 10.

Figura 10: Gráfico de PERT



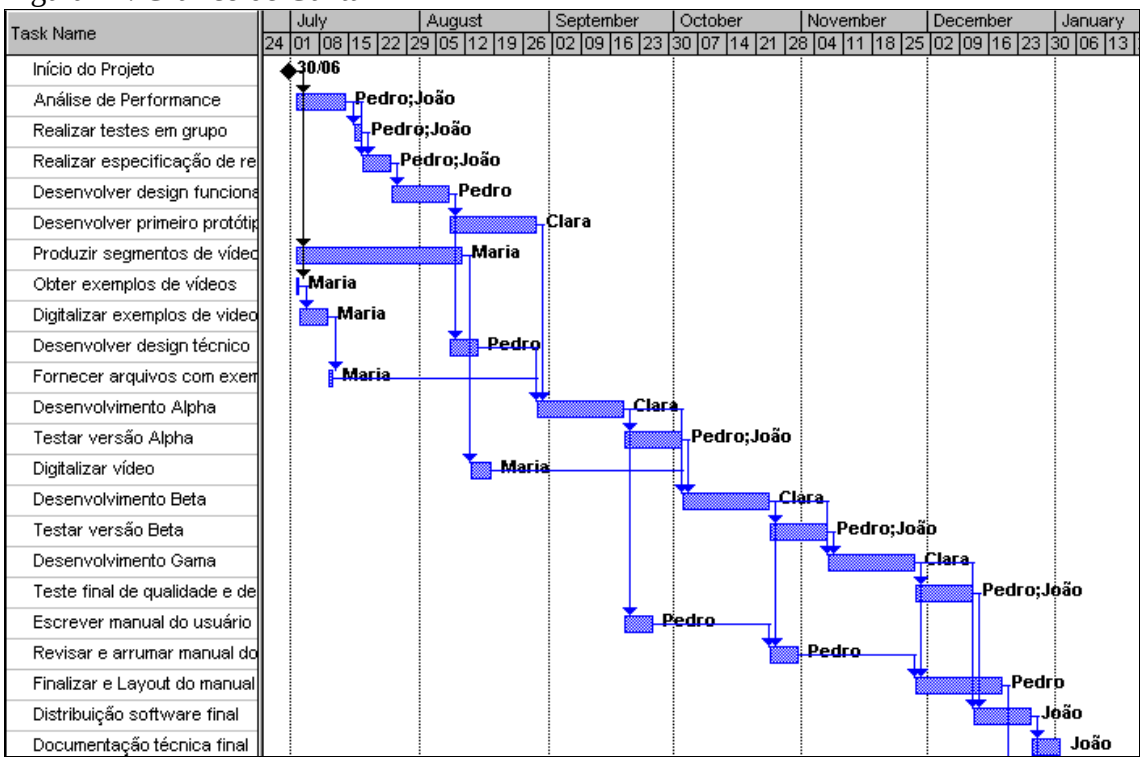
3.4.2.2 Gráfico de Gantt

Gráfico de Gantt é um tipo de gráfico de barras que graficamente ilustra quanto tempo uma tarefa levará, quando ela deve iniciar e quando deve ser concluída. Este gráfico também representa o tamanho de uma tarefa, o percentual completado de cada tarefa até o momento e quem é responsável por cada parte do projeto. (Strauss, 1997; Sommerville, 1992)

É uma ferramenta útil para traçar e rastrear o progresso das tarefas e verificar se o *schedule* está sendo cumprido.

Strauss (1997) afirma que o gráfico de Gantt é uma excelente maneira para medir o progresso das tarefas e demonstrar o que a equipe necessita fazer e quando deverá ser feito. Além disso o gráfico de Gantt permite agrupar tarefas, mostrar as tarefas de cada membro da equipe, enfim, permite agrupar as informações sob diversas formas. A Figura 11 apresenta um exemplo de gráfico de Gantt.

Figura 11: Gráfico de Gantt



3.4.3 Estrutura Organizacional

Para se organizar o gerenciamento tem-se que delegar papéis para os indivíduos e determinar responsabilidades para realizar os objetivos do projeto. A organização é basicamente motivada pela necessidade de cooperação quando os objetivos levariam muito tempo para serem realizados por uma única pessoa. Portanto, o objetivo de uma estrutura organizacional visa facilitar a cooperação em busca de um objetivo comum.

Teóricos e práticos em gerenciamento tem estudado os efeitos de diferentes estruturas organizacionais na produtividade e eficiência do grupo. Pois o objetivo da organização é encorajar a cooperação, e a cooperação depende substancialmente de características humanas.

Pode-se categorizar, segundo Ghezzi et al. (1991), a organização da equipe de desenvolvimento de software de acordo com onde se encontra o controle da tomada de decisão:

- **Organização de equipe com controle centralizado:** onde há um líder reconhecido que é responsável em desenvolver o projeto e resolver todas as questões técnicas. Deste modo, várias pessoas se reportam ao supervisor que controla diretamente suas tarefas e é o responsável pelo desempenho da equipe. O controle centralizado é baseado em uma estrutura organizacional hierárquica a qual existem vários níveis de gerência. Uma maneira de centralizar o controle do desenvolvimento de software é através de um programador chefe – responsável pelo *design* e todos os detalhes técnicos do projeto – que se reportaria ao gerente de projeto que é responsável pelos aspectos administrativos do projeto. Um ponto negativo desta categoria deve-se ao fato de que toda a comunicação e todas as decisões devem ser realizadas pelo programador chefe podendo ocasionar uma sobrecarga. Estando, desta forma, o sucesso atrelado a habilidade do programador chefe.
- **Organização de equipe com controle descentralizado:** nesta categoria as decisões devem ser tomadas em consenso pelo grupo. Um ponto negativo desta categoria deve-se ao fato dela não ser apropriada a equipes grandes, onde o excesso de comunicação pode sobrecarregar o grupo gerando redução na produtividade.

- **Organização de equipe com controle misto:** tenta combinar os benefícios do controle centralizado e descentralizado, minimizando suas desvantagens. Esta forma de organização distingue os programadores entre sênior e júnior. Cada programador sênior lidera um grupo de juniores. Sendo que os seniores se reportam a um gerente de projeto. Ou seja, o controle é investido no gerente de projeto e nos programadores seniores, enquanto a comunicação é descentralizada entre cada membro da equipe e seu supervisor.

3.4.4 Gerenciamento do Risco

De um projeto é esperado que seja produzido um software confiável, com tempo e recursos determinado. Entretanto, em qualquer projeto há o risco de não produzir o produto desejado, extrapolar o tempo e/ou orçamento. Riscos acompanham qualquer atividade humana. (Ghezzi et al., 1991)

A análise dos riscos, segundo Pressman (1995), é composta por 4 atividades distintas:

- **Identificação dos Riscos -** é possível dividir os riscos em diferentes categorias:
    - Riscos de projeto: identificam problemas orçamentários, de cronograma, de pessoal, de recursos, de clientes e de requisitos, e o impacto dos mesmos sobre o projeto de software.
    - Riscos técnicos: identificam potenciais problemas de projeto, implementação, interface, verificação e manutenção. Além disso, a ambigüidade de especificação, incerteza técnica, obsolescência técnica e tecnologia "de ponta" também são fatores de risco.
    - Riscos de negócio: são insidiosos, porque podem destruir os resultados até mesmo dos melhores projetos de software. Os 5 riscos de negócio de maior destaque são: (i) construir um excelente produto que ninguém realmente quer (risco de mercado); (ii) construir um produto que não mais se encaixe na estratégia global de produtos da empresa; (iii) construir um produto que a equipe de vendas não sabe como vender; (iv) perder o apoio da alta administração devido à mudança de enfoque ou mudança de pessoas (risco

administrativo); e (v) perder o compromisso orçamentário ou de pessoal (risco orçamentário).

- **Projeção dos Riscos –** a projeção dos riscos, também chamada estimativa dos riscos, tenta classificar cada risco de duas maneiras – a probabilidade de que o risco seja real e as conseqüências dos problemas associados ao risco, caso ele ocorra. O planejador do projeto, juntamente com outros gerentes e o pessoal técnico, executa quatro atividades de projeção dos riscos, conforme Charette (*apud* Pressman, 1995): (i) estabelecimento de uma escala que reflita a probabilidade percebida de ocorrência de um risco; (ii) delineamento das conseqüências do risco; (iii) estimativa do impacto do risco sobre o projeto e o produto; e (iv) anotação da precisão global da projeção dos riscos de forma que não haja mal-entendidos.
- **Avaliação dos Riscos –** durante a avaliação dos riscos é examinado mais detalhadamente a precisão das estimativas que foram feitas durante a projeção dos riscos, tentando determinar uma ordem de prioridade para os riscos que foram descobertos e começa-se a pensar em maneiras de controlar e/ou evitar riscos que têm a probabilidade de ocorrer. A fim de que a avaliação seja útil, um nível de risco referente deve ser definido, informa Charette (*apud* Pressman, 1995). Para a maioria dos projetos de software, o custo, os prazos e o desempenho representam três níveis de risco referentes típicos.
- **Gerenciamento e Monitoração dos Riscos –** o trio descrição , probabilidade e impacto associado a cada risco é usado como base a partir da qual os passos de gerenciamento dos riscos (ou aversão a riscos) são desenvolvidos. É importante observar que os passos de administração dos riscos acarretam custos adicionais ao projeto. Portanto, parte da administração dos riscos significa avaliar quando os benefícios advindos das atividades tomadas para evita-los são ultrapassados pelos custos associados à implantação dos mesmos. Essencialmente, o planejador de projetos executa uma análise de custo-benefício clássica.

3.4.5 Análise Conclusiva

Considerando que a fase de monitoramento e controle permanece durante toda a execução do projeto, a última fase do processo de gerenciamento (segundo a visão de

Jalote, 1997) é executada quando o processo de desenvolvimento termina. A razão básica para realizar a análise conclusiva é prover informação sobre o processo de desenvolvimento. Esta informação se faz necessária para compreender as características do processo, pois as informações retiradas desta análise darão suporte e estimativas para futuros projetos. Além disso, esta fase prevê realizar uma análise sobre o próprio processo utilizado.

3.5 Projetos de Pesquisa

Esta seção apresenta alguns modelos e ferramentas desenvolvidos recentemente, como resultado de projetos de pesquisa, para atuarem no âmbito do gerenciamento de projetos de software.

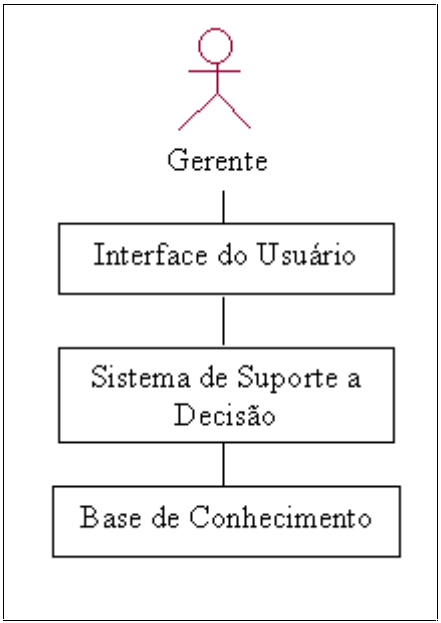
3.5.1 Prompter

Uma pesquisa apresentada por O'Connor & Jenkins (1999) e que resultou em uma tese de PhD (O'Connor, 2000) inclui agentes inteligentes para auxiliar no processo de tomada de decisão, sendo desenvolvido utilizando arquitetura cliente-servidor de natureza distribuída e multi-plataforma. O objetivo desta pesquisa vai de encontro a carência no suporte ao processo de tomada de decisão nas ferramentas de gerenciamento de projeto, desenvolvendo um modelo que encapsula o conhecimento do especialista e torna-o disponível para todos os usuários.

Os autores acreditam que os benefícios desta abordagem aplicada no processo de tomada de decisão no domínio do gerenciamento de projeto de software são: (i) as sugestões auxiliam o usuário a balancear custo, qualidade e tempo nas tomadas de decisão sobre os recursos utilizados no projeto; (ii) o conhecimento é adequado para diferentes ciclos de vida e (iii) medições são sugeridas de forma que possibilitarão ao usuário verificar se o projeto está atingindo os objetivos e replanejar os caminhos para atingi-los se necessário.

Uma visão de alto nível da arquitetura proposta nesta pesquisa pode ser visualizada na Figura 12. A arquitetura possui uma interface com o usuário,um sistema de apoio a decisão e uma base de conhecimento.

Figura 12: Arquitetura do Sistema



Fonte: O'Connor & Jenkins (1999)

Um protótipo desta ferramenta, denominada Prompter, foi desenvolvido em Java utilizando a linguagem JESS (*Java Expert System Shell*) para implementação do agente. (O'Connor & Jenkins, 1999; O'Connor, 2000)

3.5.2 CSCW e Gerenciamento de projetos

Knotts et al. (1998) apresenta uma abordagem conceitual para o gerenciamento de projetos o qual inclui CSCW (*Computer-Supported Cooperative Work*) durante o planejamento de projetos. Os autores propõem uma ferramenta a qual o planejamento de projeto seria inicialmente decomposto e distribuído para a equipe de trabalho que trabalharia independentemente e em paralelo, para determinar os requisitos das atividades que lhes foram designadas. Com relação a isto, a ferramenta seria usada para simular o projeto em grupo definindo quais planos seriam revistos, modificados e aperfeiçoados.

A ferramenta representaria o projeto como um processo dinâmico e distribuído de múltiplos agentes interagindo independentemente em busca de um objetivo comum. O propósito desta ferramenta, segundo Knotts et al. (1998), foi aplicado no campo de projeto de circuitos eletrônicos.

Considerando que o projeto tenha todas as atividades já completadas o passo seguinte iniciaria desenvolvendo uma simulação do projeto completo. Durante a simulação, agentes autônomos baseados em computador dariam a sequencia das atividades do projeto automaticamente. Para isto, a ferramenta prove um agente para cada atividade. Cada agente seria projetado para se comportar como sendo o responsável pela atividade a qual foi designado. Os agentes operariam em um ambiente *blackboard* onde todos os recursos seriam listados no quadro-negro e, desta forma, visível por todos os agentes. A simulação do projeto inteiro seria também observada pelo usuário em tempo real, pois a ferramenta executa e mostra em uma animação a representação do progresso.

3.5.3 DSPMtool

O DSPMtool, desenvolvido na School of Computer Science and Engineering da University of New South Wales, é um sistema de gerenciamento de projeto de software distribuído (cliente/servidor) que consiste de ferramentas e técnicas usadas para coletar, analisar, integrar e disseminar as saídas dos vários processos de gerenciamento de projeto de software. (Surjaputra & Maheshwari 1999)

A idéia central do DSPMtool é gerenciar projetos de software considerando os produtos do processo de desenvolvimento de software, tais como projeto, especificação de requisitos, componentes de software. Para o DSPMtool os documentos podem ser, por exemplo, arquivos multimídia, código fonte, executáveis, documentos texto, dentre outros. As características de gerenciamento de configuração do DSPMtool fornecem a infraestrutura para gerenciar os vários elementos produzidos durante o desenvolvimento de um projeto de software distribuído. Surjaputra & Maheshwari (1999) acreditam que com esta característica, o gerente de projeto controla como os documentos são distribuídos para os membros da equipe tornando-se uma maneira de planejar e controlar atividades, cronograma, recursos e custos envolvidos no desenvolvimento de projeto de software.

3.5.4 SPPA

O SPPA (*Software Project Planning Associate*), desenvolvido na linguagem de programação Java e acesso através dos *browsers*, é projetado para auxiliar ao gerente de

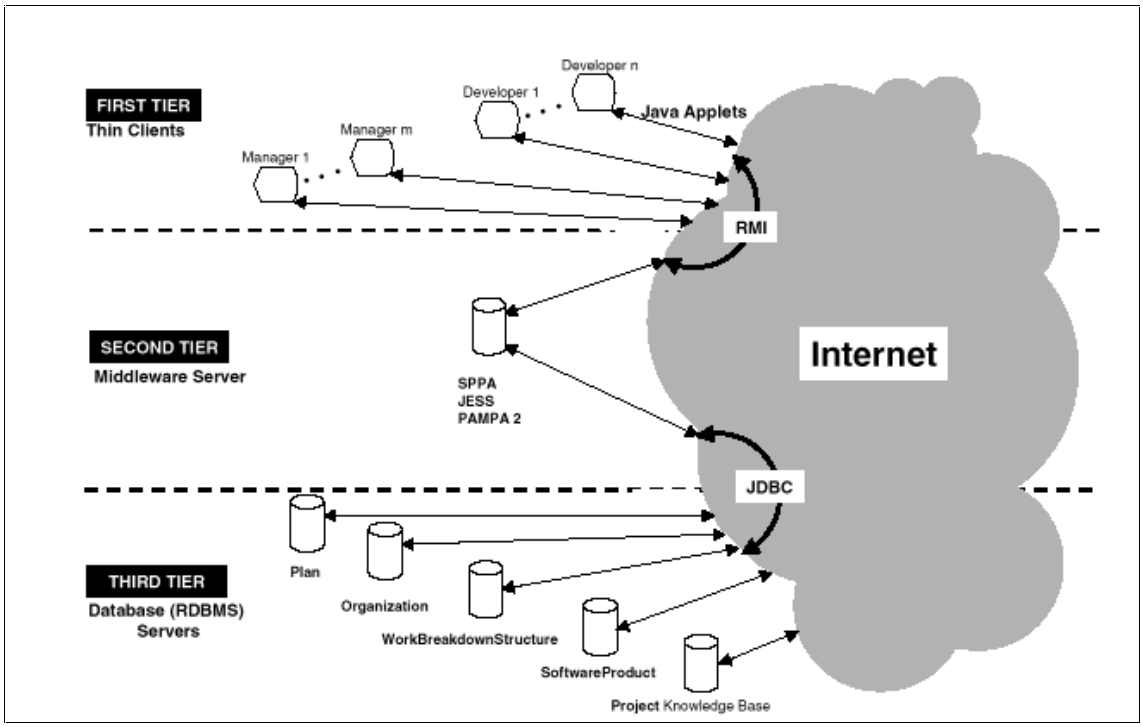
projeto de software a inicializar o planejamento de projeto, refinar o planejamento, organizar a equipe, definir o cronograma, medir, visualizar, monitorar e controlar, prognosticar e coletar informaçãoes. (Wu & Simmons, 2000)

Os recursos, tarefas, cronograma e *milestones* do projeto são descritos no planejamento. Conforme o processo de desenvolvimento evolui medidas do projeto do software são reunidas e reportadas. Prognósticos do processo de software são efetuados e recomendações são dinamicamente reportados sugerindo o desenvolvimento de software que deve ser executado para cumprir com o planejado.

O SPPA, segundo Wu & Simmons (2000), foi desenvolvido de acordo com um modelo de Planejamento Baseado em Conhecimento que permite ao gerente acompanhar o projeto. Um agente de planejamento de projeto que reporta problemas e sugestões para o gerente auxiliando na garantia que o projeto satisfaz o orçamento e o tempo previsto.

A arquitetura do SPPA consiste em uma arquitetura de 3 camadas, conforme visualizado na Figura 13.

Figura 13: Arquitetura 3 camadas



Fonte: Wu & Simmons (2000)

A primeira camada contém o *"thin client"* que é a interface com o gerente e desenvolvedores. Nesta camada faz-se necessário somente um *browser web* e a máquina virtual Java. A comunicação com a segunda camada dá-se através de *Remote Method Invocation* (RMI).

A segunda camada abriga o *Middleware serve* onde está operando PAMPA 2 (ambiente que utiliza um sistema especialista para criar agentes inteligentes), JESS (linguagem para programação de agentes) e SPPA. O SPPA é um subsistema que permite: (i) planejamento (definição de atividades, ciclo de vida, *milestones*, gráfico de Gantt,...); (ii) agentes inteligentes de planejamento (auxilia o gerente a acompanhar o *status* do projeto); (iii) base de conhecimento (contém as regras e fatos para os agentes inteligenes operarem); (iv) informação do grupo (mantém informações referentes aos custos de pessoal, que constitui a maior despesa de um projeto). A comunicação com a terceira camada ocorre através de JDBC (*Java Database Connectivity*). (Wu & Simmons 2000)

A terceira camada, servidora de banco de dados, utiliza o sistema de gerenciamento de banco de dados relacional IBM DB2.

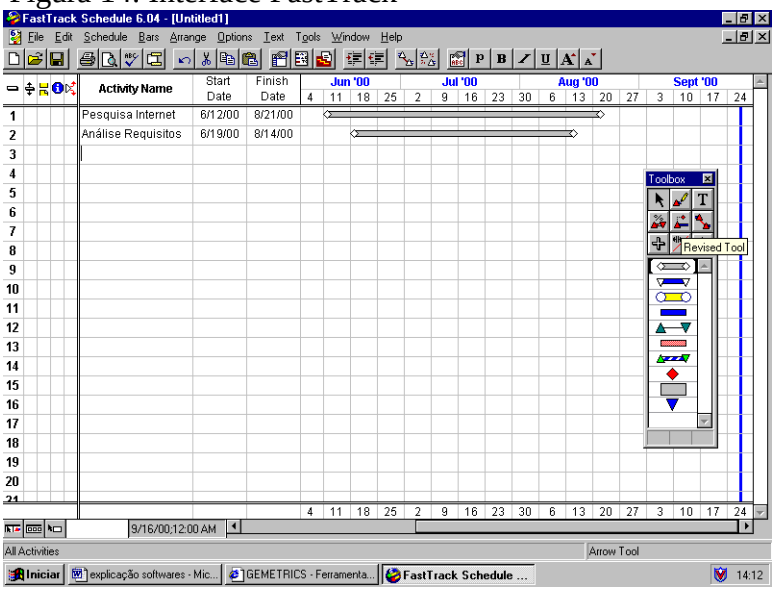
3.6 Ferramentas de Gerenciamento de Projeto

Esta seção apresenta uma descrição sucinta de algumas ferramentas de gerenciamento de projeto, sendo estas aplicadas no escopo de qualquer projeto (ou seja, não são específicas para projeto de software). Este estudo foi efetuado visando analisar as principais características das ferramentas existentes atualmente e, conseqüentemente, embasar a proposta apresentada. Vale ressaltar que esta lista refere-se apenas a um pequeno conjunto de ferramentas, existindo ainda diversas outras disponíveis.

3.6.1 FastTrack

O FastTrack é um software de gerenciamento de projetos, cuja interface é ilustrada na Figura 14. Este software foi desenvolvido pela AEC software. O FastTrack 6.04a é a última versão do sistema, tendo sido desenvolvido em 1999, sendo que, para a análise foi utilizada uma versão demo cuja única limitação é a não possibilidade de salvar um projeto em disco.

Figura 14: Interface FastTrack



O FastTrack tem como objetivo manter o histórico de projetos, atividades, tarefas e prazos finais. No seu nível mais básico, o FastTrack é simplesmente um gráfico de linha ao longo do qual são colocadas barras de atividade para representar os começos e fins destas atividades, sendo este gráfico conhecido como Gráfico de Gantt.

Desta forma, o FastTrack automatiza a construção do gráfico de Gantt, refletindo o começo e encerramento de atividades de forma gráfica. Quando uma determinada linha/atividade é alterada no gráfico, automaticamente, dados como o custo de cada atividade e o valor total do projeto são alterados. Pode-se escolher entre 17 tipos diferentes de barras de atividade para denotar informações diferentes.

A qualquer instante é possível mudar o gráfico de Gantt para ver um maior ou menor período de tempo, mudança estas disponibilizadas através das unidades para exibição, que podem ser: tempo em horas, dias, semanas, meses, trimestre, ou anos, no exemplo acima (Figura 14), pode-se notar que o software está usando o período em semanas.

Em um primeiro instante as entradas de informações são textuais, ou seja, o nome da tarefa, data de início, data de fim entre outras informações são incluídas na base, após serem inseridas, pode-se trabalhar com as datas de início e fim das tarefas diretamente no Gráfico de Gantt. Cada tarefa contém uma barra que representa o período da atividade, podendo também esboçar as atividades em níveis diferentes, ou seja, estruturar tarefas de forma hierárquica.

Além disso, pode-se visualizar através do gráfico quando uma atividade foi revisada, o andamento da tarefa e quando foi completada de fato. Refletindo desta forma, a relação entre o planejamento inicial e a execução efetiva do projeto.

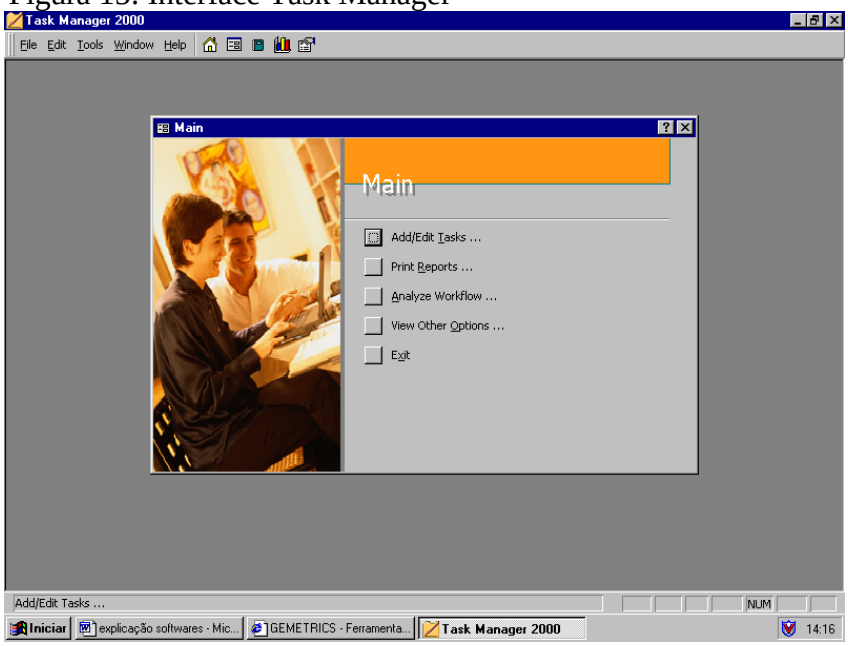
Outro ponto positivo a ser ressaltado é a capacidade de se criar *scripts* para automatizar tarefas, ou seja, pode-se agrupar uma lista de ações a serem executadas em um único arquivo. Executando este arquivo, as ações são automaticamente cumpridas.

Como ponto negativo, pode-se ressaltar a falta de gerenciamento/controle do software com relação aos *links* entre tarefas, ou seja, pode-se criar *links* incorretos entre tarefas e fases.

3.6.2 Task Manager

O Task Manager é um software de gerenciamento de projetos, podendo sua interface ser visualizada na Figura 15. Este software foi desenvolvido pela Orbisoft, cujo site encontra-se em http://www.orbisoft.com. A versão pesquisada foi desenvolvida em 2000 e, atualmente, encontra-se na versão 1.00.729, sendo que, para a análise foi utilizada uma versão demo, sem nenhuma restrição/limitação.

Figura 15: Interface Task Manager



O Task Manager 2000 foi especialmente projetado para ajudar a organizar e administrar todos os trabalhos e tarefas de maneira simples, seguindo o padrão dos *wizards.* Pode ser usado pessoalmente ou em um ambiente de equipe, podendo

administrar pessoal e compartilhar tarefas, trabalhos, e projetos. Este compartilhamento é efetuado através da disponibilização do banco de dados em um único computador na rede, através do qual os outros elementos da equipe acessam este computador para usufruir as informações pertinentes a um determinado projeto.

Este software permite adquirir uma visão rápida de todas as tarefas, controlar prazos finais, equilibrar automaticamente as cargas de trabalho, previsões de gargalos de trabalho e tempos. Além disso, o Task Manager 2000 calcula automaticamente o custo total de desenvolvimento do projeto, com base nos custos de cada pessoa, alocada para participar do projeto.

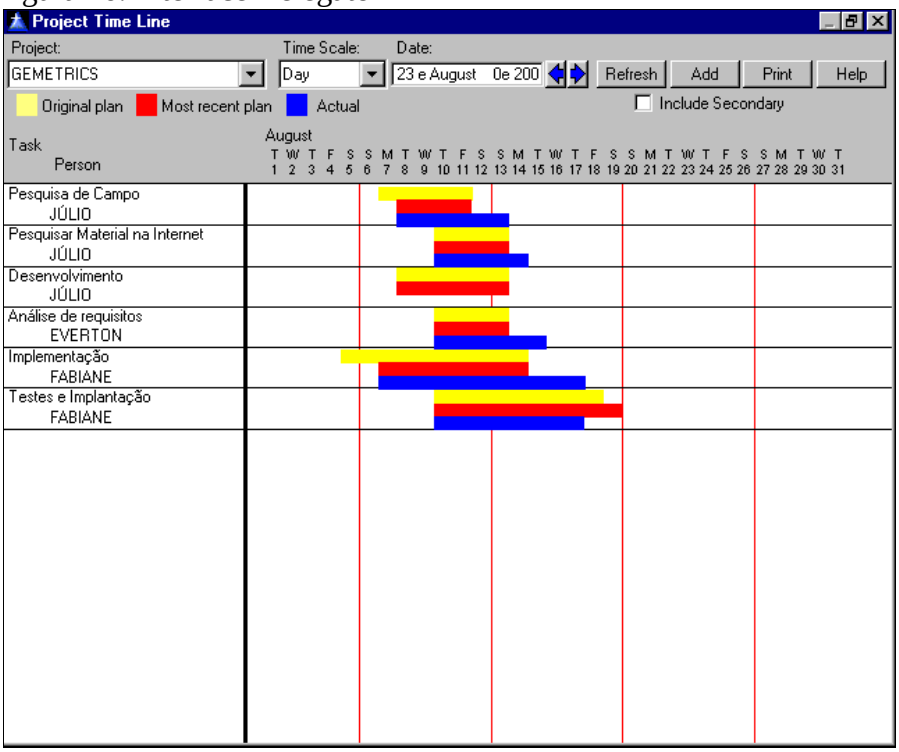
Possui uma interface de fácil manuseio, mas é deficiente na parte de relatórios/gráficos, onde possui alguns gráficos não parametrizados (não se tem opções de escolher datas, o software demonstra sempre os últimos 5 anos) sobre alocação de trabalho, custo/mês, tarefas/mês, entre outros. Não aborda gráficos significativos, como por exemplo, o Gráfico de Gantt ou Gráfico de PERT, gráficos típicos de gerenciamento de projetos, outro ponto negativo é sua lentidão para troca entre janelas. Possui um *help online*, para dirimir qualquer dúvida sobre a aplicação.

3.6.3 Delegator

O Delegator é mais um software de gerenciamento de projetos, cuja interface pode ser visualizada na Figura 16. Este software foi desenvolvido pela Madrigal Soft Tools Inc, com sede no Canadá, sendo o seu site encontrado em http://www.madrigalsoft.com. A versão pesquisada foi desenvolvida em 2000 e, atualmente, encontra-se na versão 3.0, sendo que, para a análise foi utilizada uma versão demo, sendo limitada para uso de apenas um usuário.

Delegator foi desenvolvido especialmente para gerentes e demais pessoas que coordenam e administram as atividades de outras pessoas. O Delegator ajuda a localizar as diversas tarefas dadas as pessoas, cargas de trabalho do pessoal, comunicar prioridades, desempenho de pessoal com o passar do tempo e administrar projetos incluindo muitas tarefas e envolvendo várias pessoas.

Figura 16: Interface Delegator



Como uma de suas características pode-se destacar o gráfico de Gantt que é utilizado para visualizar as tarefas, referenciando os recursos humanos alocados, sendo distribuídas por períodos de tempo podendo-se visualizar o gráfico por período de dias, semanas, meses ou trimestre. Além disso, pode-se destacar o *help on-line* que auxilia no entendimento da ferramenta.

Como ponto principal do software pode-se ressaltar os três planos do cronograma que são dispostos no gráfico de Gantt, conforme pode ser visualizado na Figura 16, ou seja, a barra em amarelo refere-se as datas do plano inicial traçado pelo gerente para uma determinada tarefa, a barra vermelha refere-se ao plano mais recente traçado pelo gerente, e por fim a barra de cor azul, faz referência ao cronograma realmente traçado pela pessoa ou grupo responsável pela tarefa. Através destes três planos é possível fazer um comparativo do andamento das tarefas, da concepção até a conclusão da atividade.

Através desta ferramenta é possível gerar relatórios referentes ao histórico de uma determinada pessoa, histórico de um grupo de pessoas, histórico de um projeto ou de vários projetos, além de listar tarefas, entre outros. Todos os relatórios são

parametrizados por datas a serem definidas pelo usuário, tornando assim, uma ferramenta de trabalho mais flexível.

Como pontos negativos se pode ressaltar a falta de interatividade do gráfico de Gantt, pois as tarefas dispostas no gráfico só podem receber entradas de dados de maneira textual. Além disto, pode-se destacar a falta de dependência entre as tarefas de um projeto, ou seja, não se pode vincular uma tarefa a outra (estabelecer relação de precedência).

3.6.4 Alexys Team

Alexsys Team é um sistema de administração de equipe para Windows que organiza o trabalho de uma equipe ou organização. Através da utilização desta ferramenta, em uma organização, os gerentes podem administrar o trabalho do grupo de forma que:

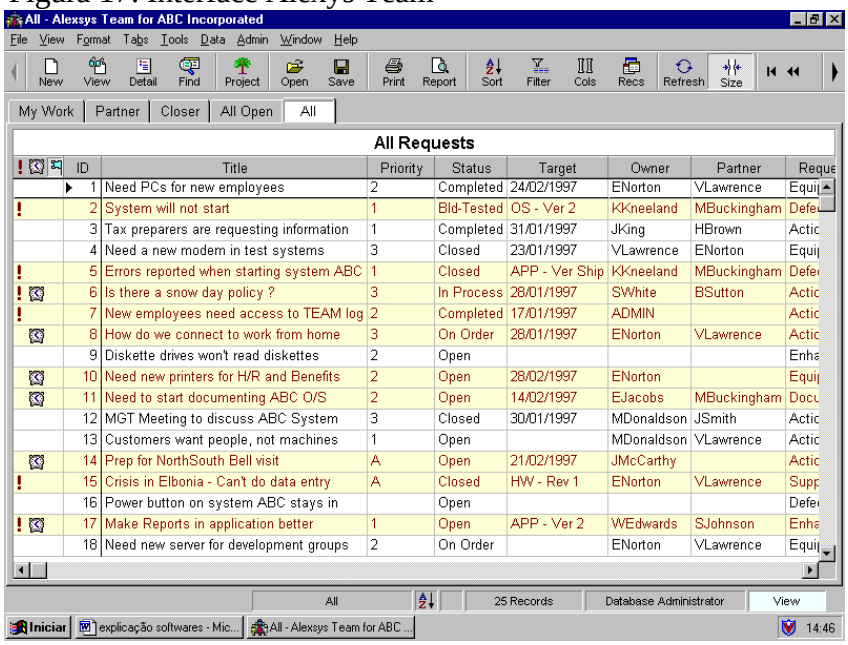
- o grupo pode nomear tarefas coletivamente entre eles, ou seja, formar grupos para executarem uma tarefa;
- os sócios da organização podem interagir e podem tomar decisões;
- o grupo pode descobrir quando uma tarefa não está procedendo dentro do planejado.

Na instalação desta ferramenta são implantados dois bancos de dados: um banco de dados novo e um banco de dados de amostra. O banco de dados novo permite aos gerentes da organização registrar o trabalho de seus subordinados, enquanto o banco de dados de amostra provê exemplos de como a ferramenta pode ser usada, através do qual o usuário poderá se embasar para implementar o seu banco de dados.

O Alexys Team pode ser visualizado na Figura 17.

As informações pertinentes às atividades são previamente cadastradas em tabelas anexas, para posteriormente serem referenciadas dentro da estrutura das atividades, ou seja, cada item da estrutura de uma atividade, possui várias opções pré-definidas para preenchimento, tornando o trabalho mais fácil e rápido. Além disto, é possível passar as informações de uma tarefa via e-mail para outra pessoa envolvida no projeto.

Figura 17: Interface Alexys Team



Como ponto positivo se pode destacar a flexibilidade de configuração do ambiente de trabalho (cores do ambiente, informações a serem dispostas na tela, ordem de disposição destas informações, fontes dos textos), ou seja, através de alguns comandos simples é possível adaptar a área de trabalho ao estilo do usuário.

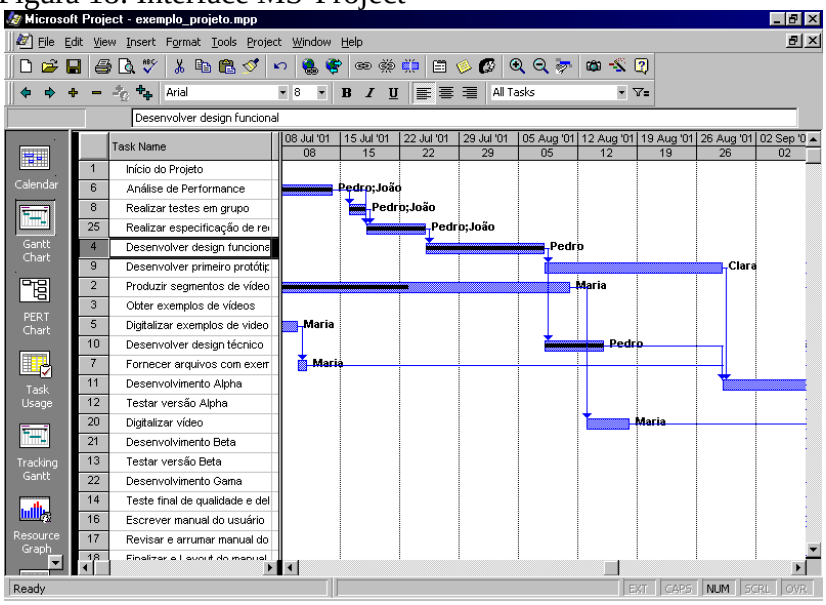
Outro ponto positivo a ser ressaltado é a possibilidade da criação de gráficos e relatórios, ou seja, pode-se montar um gráfico ou relatório com qualquer informação que esteja armazenada no banco de dados do sistema, tornando a ferramenta mais flexível. Além disso, pode-se optar por vários estilos de gráficos para visualização destas informações, por exemplo, gráfico de pizza, gráfico de linha, gráfico 3D, gráfico de área entre outros. Podendo-se também salvar os gráficos e os relatórios em arquivos anexos para futuras comparações.

Como ponto negativo se pode ressaltar a falta de um gráfico para gerenciamento de tarefas e cronograma, como por exemplo, gráfico de Gantt e/ou gráfico de PERT, pois através destes gráficos é possível melhor visualizar a distribuição das tarefas com relação ao tempo do projeto, tarefas estas que só são listadas textualmente.

3.6.5 MS-Project

MS-Project é uma ferramenta para gerenciamento, organização e controle de projetos. Para análise foi utilizada a versão 98 completa. A interface do Ms-Project pode ser visualizada na Figura 18.

Figura 18: Interface MS-Project



A principal função do Ms-Project é gerenciar as tarefas em função do tempo. Utiliza o gráfico de Gantt como o principal gráfico para visualizar tarefas. Através das linhas pode-se inserir e modificar as descrições, durações, datas de início, datas de término e outras informações sobre as tarefas. As barras estão preparadas para exibir graficamente as durações das tarefas em uma escala de tempo.

Para um modo diferente de visualização das tarefas pode-se utilizar o gráfico de PERT, que possui maior ênfase nas interdependências entre as tarefas. O software disponibiliza um gráfico de recursos, sendo possível através deste verificar se não há alguém super alocado, ou seja, se algum recurso é atribuído para trabalhar mais horas em determinado período de tempo do que está disponível em seu calendário, podendo assim, manter bem distribuídas as tarefas entre os seus funcionários.

Este software implementa a funcionalidade de efetuar previsões e evitar problemas de agendamento antes que eles ocorram, considerando as tarefas que ameaçam “estourar” o orçamento e/ou conflitos de agendamento que possam ultrapassar o prazo do projeto.

Pode-se utilizar a linha de base (*baseline*) que contém estimativas originais de custos, recursos e agendamento. Ao comparar as estimativas de sua linha de base com as estimativas “atuais”, pode-se efetuar as alterações necessárias no plano do projeto.

O software possui recurso de *hyperlink*, onde o navegador da web é iniciado localizando o endereço web especificado. Pode-se publicar informações do projeto em estudo em formato HTML e seus gráficos em formato GIF.

Para receber e enviar mensagens do grupo de trabalho, os integrantes da equipe usam caixa de entrada eletrônica. A caixa de entrada é utilizada pelo gerente do grupo de trabalho para receber e armazenar respostas às mensagens de atribuição, atualização e *status* que são enviados aos integrantes da equipe. Essas caixas de entrada devem estar conectadas através de correio eletrônico, da Web ou rede interna (intranet ou web interna). O grupo de trabalho poderá usar mensagens especiais de atribuição, atualização e *status* do MS-Project para:

- Atribuir tarefas
- Aceitar ou recusar uma atribuição de tarefa
- Solicitar e submeter relatórios de status
- Enviar e receber atualizações de tarefas

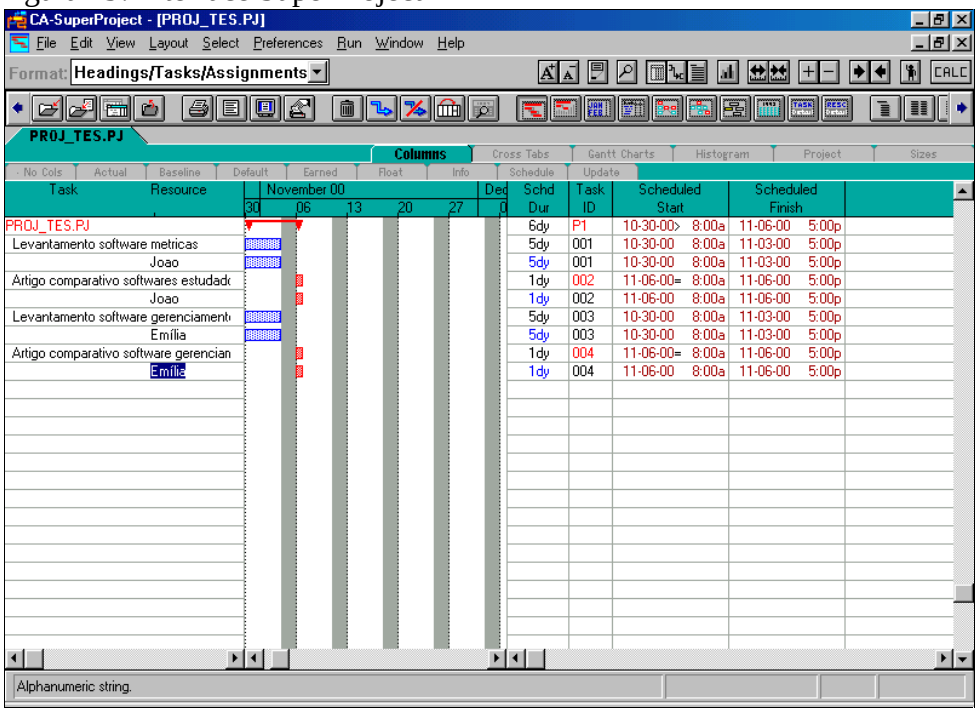
Outro ponto positivo do MS-Project é a flexibilidade na geração de relatórios, que compreendem um conjunto predefinido de informações detalhadas sobre o projeto, o MS-Project possui mais de 20 relatórios predefinidos.

3.6.6 SuperProject

SuperProject é uma ferramenta destinada ao gerenciamento de projetos, produzida pela Computer Associates, cuja interface pode ser visualizada na Figura 19. O software pesquisado refere-se a versão 4.0, e não possui nenhuma restrição a sua utilização.

O SuperProject é constituído de dois módulos. Um denominado *Application Program Interface* que constitui o módulo principal do programa, onde encontra-se a interface para o gerenciamento de projetos. O segundo módulo, denominado *CA-Realizer* contem um linguagem de desenvolvimento para criar macros e aplicações.

Figura 19: Interface SuperProject



Este software permite o gerenciamento de recursos, custo e tempo de projetos, através de gráfico de Gantt e gráfico de PERT. Possui também a possibilidade do usuário iniciar a criação de um projeto a partir de um *template* disponível. Quanto aos relatórios, o SuperProject mostrou-se bastante versátil na construção dos mesmos.

No que se refere as variáveis utilizadas, o SuperProject não demonstrou nenhuma restrição. Ou seja, pode-se configurar a unidade de tempo que se deseja trabalhar (hora, dia, mês,...), o calendário (dia e hora) de cada recurso vinculado ao projeto, dentre outros.

O SuperProject trabalha com diferentes visões do projeto, permitindo que se visualize os recursos utilizados, ou os custos envolvidos, ou detalhamento das tarefas, dentre outros.

Como item que o difere dos demais, está sua capacidade de trabalhar com inclusão de outros projetos ou construir *links* com projetos externos. A diferença reside no fato que se optando pela inclusão, o sistema anexará todo o projeto selecionado como parte integrante de um mesmo projeto. No segundo caso, é permitida a inserção de um *link* (vincular duas tarefas) entre projetos distintos. No entanto, esta operação é permitida somente para projetos residentes em uma mesma máquina.

3.7 Análise Comparativa das Ferramentas de Gerenciamento de Projetos

Com base nas ferramentas de gerenciamento de projeto apresentadas, constatou-se que a maioria destas ferramentas trabalha com o gráfico de Gantt, sendo o gráfico de PERT implementado em menor escala. O recurso de *baseline* é implementado por apenas duas das ferramentas: MS-Project e Delegator.

No que tange os relatórios pode-se destacar a flexibilidade do MS-Project que oferece inúmeros relatórios aos usuários, sendo que, o software Alexys Team também oferece bastantes opções neste item.

Vale ressaltar que todas as ferramentas apresentam um *help* para auxílio ao "uso" da ferramenta, no entanto, destaca-se que o MS-Project expande o *help* para o esclarecimento de conceitos vinculados ao gerenciamento de projeto.

Com relação ao suporte de trabalho em grupo ou projeto sendo desenvolvido de forma distribuída apenas duas ferramentas apresentam algum suporte com este fim. Sendo uma delas o MS-Project que permite salvar o projeto em HTML para disponibilizar na Web e tem suporte a trabalho em equipe (utilizando recurso de *e-mail*, sem uma visão distribuída). A outra ferramenta é o Alexys Team que visa o gerenciamento do trabalho em equipe, mas assim como o MS-Project, considerando que a aplicação não está sendo desenvolvida de maneira distribuída.

3.8 Considerações Finais

O gerenciamento de projetos é uma tarefa complexa que envolve a análise de diversos fatores, tais como tempo, recursos e tarefa. Diversos são os instrumentos existentes para auxiliar ao gerente dentre eles pode-se citar os gráficos de PERT e Gantt, bem como, as ferramentas apresentadas.

O sucesso do projeto está fortemente atrelado ao efetivo gerenciamento do processo tornando, portanto, indispensável o uso dos instrumentos citados pelo gerente.

Sendo que o principal objetivo de um gerenciamento de projeto é a busca pela qualidade do produto e produtividade da empresa/pessoal, e considerando a frase citada

por Braga (1996) que afirma não se poder gerenciar o que não se pode medir, a seção seguinte apresenta algumas formas de se mensurar um projeto de software.

4 MÉTRICA DE SOFTWARE

Na área de engenharia, a medição tem sido um aspecto de grande importância, sendo que se pode desfilar uma fila interminável de grandezas as quais sofrem este tipo de tratamento: dimensões físicas, peso (ou massa), temperatura, tensões e correntes elétricas, dentre outros. Na computação, alguns parâmetros são quantificados como forma de expressar as potencialidades de determinadas máquinas, tais como a capacidade de um processador de executar um certo número de instruções por segundo (MIPS), as capacidades de armazenamento (Mbytes), a freqüência do *clock* do processador (MHz), dentre outros.

A engenharia de software é uma coleção de técnicas que aplicam uma abordagem de engenharia para construção e suporte ao produto de software. A engenharia de software envolve atividades de gerenciamento, custo, planejamento, modelagem, especificação, projeto, implementação, teste e manutenção (conforme descrito no capítulo 2). Por abordagem de engenharia compreende-se que cada atividade é entendida e controlada, ocorrendo poucas surpresas na especificação do software, projeto, construção e manutenção. Considerando que a ciência da computação fornece fundamentação teórica para construção do software, sendo a engenharia de software focada na implementação de software de maneira controlada e científica. (Fenton & Pfleeger, 1997)

Fenton & Pfleeger (1997) lista os tipos de informações necessárias para entender e controlar um projeto de desenvolvimento de software, sobre a perspectiva do gerente e do desenvolvedor. Como um dos enfoques do presente trabalho é a utilização de métrica para o gerenciamento de projetos, abaixo se encontram as informações úteis a um gerente de projeto de software:

- Medir o tempo e esforço envolvido nas diferentes fases do processo de produção de software. Por exemplo, pode-se identificar o custo da especificação de requisitos, o custo do projeto do sistema e o custo de implementação e teste do software. Podendo-se assim identificar não somente o custo total do projeto, mas também o custo envolvido em cada etapa.

- Medir o tempo levado pela equipe para especificar o sistema, projetá-lo, implementá-lo e testá-lo, determinando assim a produtividade da equipe em cada atividade. Esta informação é útil quando mudanças são solicitadas. O gerente pode utilizar a produtividade para estimar o custo e duração das alterações.
- Avaliar a qualidade do software desenvolvido. Armazenando as falha, erros e mudanças pode-se medir a qualidade do software permitindo comparar diferentes produtos, predizer o efeito de alterações e avaliar o efeito da aplicação de novas práticas.
- Medir a funcionalidade determinando se todos os requisitos foram implementados apropriadamente. Podendo-se também avaliar a usabilidade, confiabilidade, tempo de resposta, dentre outras características para garantir a satisfação do cliente.
- Medir o tempo para executar cada atividade principal do desenvolvimento, e calcular seu efeito na qualidade e produtividade. Então pode-se avaliar a relação custo/benefício de cada prática. Viabilizando assim testar várias práticas, medir os resultados e decidir qual a melhor, por exemplo, pode-se comparar 2 métodos de projeto e verificar qual apresenta maior qualidade no código.

Esta listagem mostra, segundo Fenton & Pfleeger (1997), como a medição faz-se importante para 3 atividades básicas. Primeiro, existe medidas que auxiliam entender o que está ocorrendo durante o desenvolvimento e manutenção. As medidas tornam os aspectos do processo e produto mais visíveis, dando um melhor entendimento do relacionamento entre atividades .

Segundo, a medição permite controlar o que está acontecendo nos projetos. Por exemplo, pode-se monitorar a complexidade dos módulos e efetuar uma rigorosa revisão somente naqueles que excedem uma certa medida.

Terceiro, medições encoraja a melhorar o processo de desenvolvimento e o produto. Por exemplo, pode-se aumentar a quantidade ou tipos de revisão de projetos baseado nas medidas obtidas na especificação de requisitos.

Pressman (1995) e Braga (1996) acrescentam diversas razões para se considerar as medições um item de importância para o gerenciamento:

- quantizar a qualidade do software como produto;
- avaliar a produtividade dos elementos envolvidos no desenvolvimento do produto;
- estimar/medir o tamanho de um sistema antes de desenvolvê-lo;
- avaliar os benefícios de métodos e ferramentas para o desenvolvimento de software;
- formar uma base de dados para as estimativas;
- justificar o pleito e aquisição de novas ferramentas e/ou treinamento adicional para membros da equipe de desenvolvimento.

4.1 Classificação

De forma análoga a outras grandezas do mundo físico, as medições de software podem ser classificadas em duas categorias principais, segundo Bomfim et al.(2000) e Pressman (1995):

- as **medições diretas**, por exemplo, o número de linhas de código (LOC) produzidas, o tamanho de memória ocupado, a velocidade de execução, o número de erros registrados num dado período de tempo, dentre outros.
- as **medições indiretas**, as quais permitem quantizar aspectos como a funcionalidade, complexidade, eficiência, manutenibilidade, dentre outros.

As medições diretas, tais quais aquelas exemplificadas acima, são de obtenção relativamente simples, desde que estabelecidas as convenções específicas para isto. Por outro lado, aspectos como funcionalidade, complexidade e eficiência são bastante difíceis de quantizar.

As medições de software podem ser organizadas em outras classes, as quais serão definidas a seguir, conforme citado por Bomfim et al.(2000) e Pressman (1995):

- **métricas da produtividade**, baseadas na saída do processo de desenvolvimento do software com o objetivo de avaliar o próprio processo;
- **métricas da qualidade**, que permitem indicar o nível de resposta do software às exigências explícitas e implícitas do cliente;

- **métricas técnicas**, nas quais encaixam-se aspectos como funcionalidade, modularidade, manutenibilidade, dentre outros.

Sob uma outra ótica, definida por Pressman (1995), é possível definir uma nova classificação das medições:

- **métricas orientadas ao tamanho**, baseadas nas medições diretas da ES. Esta classe abrange todas as possíveis medidas obtidas diretamente do software. Um exemplo é a utilização de linhas de código, descrita na seção 4.2.
- **métricas orientadas à função**, que oferecem medidas indiretas. Esta classe é baseada em medidas indiretas do software e do processo utilizado para obtê-lo. Esta métrica leva em conta aspectos como a funcionalidade e a utilidade do programa. Uma abordagem proposta nesta classe é a do ponto de função (*function point*), descrita na seção 4.3.
- **métricas orientadas às pessoas**, as quais dão indicações sobre a forma como as pessoas desenvolvem os programas de computador.

4.2 Linhas de Código

As primeiras tentativas de se medir o tamanho de um sistema, conforme Braga (1996), levou em consideração as LOCs (*Lines Of Code*- Linhas de Código). É baseada na idéia de que “certo sistema possui um número maior de LOCs que um outro, portanto, é maior e mais complexo”.

Segundo Pressman (1995), os que discutem o uso desta informação, alertam para a forte dependência da linguagem no uso desta técnica. Além disso, a métrica orientada ao tamanho, classificação da LOC, tende a penalizar os programas bem estruturados e que tenham feito economia de software. É uma métrica que, na verdade, deixa de ser precisa principalmente por causa dos diversos novos fatores introduzidos pelas linguagens de programação mais recentes, além de ser de difícil aplicação para efeito de estimativas, uma vez que é necessário descer a um nível de detalhe que não é desejável na etapa de planejamento do projeto.

Os que defendem o uso desta métrica, afirmam que é um aspecto de fácil obtenção e, portanto, de fácil aplicação a qualquer projeto de software, além de destacar a quantidade de informação existente com base nesta métrica.

4.3 Análise de Pontos de Função

A técnica de Análise de Pontos de Função (FPA – *Function Point Analysis*) mede uma aplicação através das funções desempenhadas para/e por solicitação do usuário final. Sendo baseada na visão do usuário, a FPA é independente de tecnologia e pode ser utilizada para estimativas.

A técnica de Análise de Pontos de Função mede "o que" é o sistema e não "como" será, ou foi, desenvolvido. Um dos principais conceitos relativos a APF é que as funções devem ser contadas a partir da perspectiva do usuário e não do analista ou programador.

Pontos de função (*function point*), é baseada em medidas indiretas sobre a complexidade do software, cujo grupo responsável pela padronização denomina-se IFPUG (International Function Point Users Group, 2000).

A técnica, segundo IFPUG (2000), divide as funções em dados e transações, que são compostos pelos seguintes tipos:

- Funções de dados
  - Arquivos Lógicos Internos (ALI)
  - Arquivos de Interface Externa (AIE)
- Funções de transações
  - Entradas Externas (EE)
  - Saídas Externas (SE)
  - Consultas Externas (CE)

A contagem das funções do sistema através desta perspectiva torna a técnica independente de linguagem de programação, banco de dados e da experiência do programador.

4.3.1 Objetivos/Benefícios da Análise de Pontos de Função

Conforme cita Braga (1996), “pontos de função é apenas uma unidade-padrão da área de informática, mas que associada a outras medidas lhe permite definir, criar e monitorar indicadores de qualidade, produtividade e financeiros.”

Uma vez computados, os pontos por função podem ser utilizados para os mais variados fins, conforme lista Braga (1996):

- Validação de ferramentas de software, tecnologias e métodos através de comparação de resultados.
- Identificação das práticas com melhor desempenho visando a sua propagação na empresa.
- Melhorar a gerência de projetos.
- Qualificar níveis de performances atuais, em termos de produtividade, qualidade e custos.
- Medir a satisfação do usuário em relação às soluções desenvolvidas e o processo empregado.
- Quantificar a contribuição da informática para atingir os objetivos empresariais.
- Melhorar as estimativas de projetos, permitindo prever com maior acurácia e em fases iniciais do ciclo de desenvolvimento de sistemas.
- Justificar necessidades de pessoal e recursos.
- Melhorar o processo de desenvolvimento e manutenção, através da análise de pontos fortes, pontos fracos, causas de problemas e custos das falhas.
- Melhorar o relacionamento da informática com os usuários finais dos diversos departamentos.
- Melhorar a qualidade dos contratos de terceirização.
- Consiste em uma ferramenta para determinar o tamanho de um pacote a ser adquirido através da contagem dos pontos de função que ele possui.

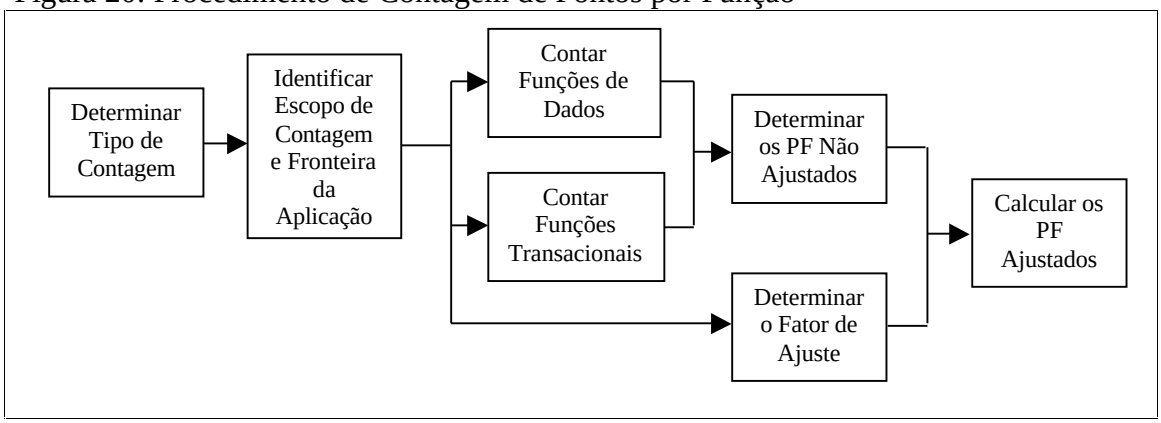
- Uma ferramenta de auxílio para determinar os benefícios que um pacote aplicacional pode oferecer a uma empresa, através da contagem dos pontos de função que refletem suas necessidades.
- Um veículo para medição de estimativas de custo e recursos requeridos para o desenvolvimento e manutenção de software.
- Um fator de normalização na comparação entre softwares.
- Acompanhamento da qualidade e produtividade visando à otimização do processo de desenvolvimento de sistemas.
- Uma ferramenta para auxiliar a decisão entre a compra de um pacote ou o desenvolvimento do aplicativo na empresa.

4.3.2 Processo de Contagem

O processo de contagem dos pontos de função, conforme Braga (1996) e Hazan (2000), constitui-se das seguintes etapas (Figura 20):

1. Determinação do Tipo de Contagem
2. Fronteiras da Aplicação
3. Funções do Tipo Dados
4. Funções do Tipo Transação
5. Determinação dos Pontos de Função Brutos
6. Determinação do Fator de Ajuste
7. Cálculo dos Pontos de Função Ajustados

Figura 20: Procedimento de Contagem de Pontos por Função



Fonte: Hazan (2000)

As seções seguintes detalham as atividades de cada uma das etapas relacionadas.

4.3.2.1 Determinação do tipo de contagem

O primeiro passo no processo de contagem de pontos de função é identificar o tipo de contagem desejada, sendo que os tipos previstos, segundo Braga (1996), são:

- Contagem em projetos de desenvolvimento: este tipo prevê a contabilização das funções identificadas no modelo lógico do sistema permitindo uma estimativa dos recursos tempo e pessoal necessário ao desenvolvimento do sistema.
- Contagem em projetos de manutenção: consiste na medição das modificações que envolvem a inclusão, alteração ou exclusão de funções.
- Contagem de uma aplicação: trata-se da aplicação da técnica de pontos de função em um sistema já totalmente desenvolvido.

4.3.2.2 Fronteiras da Aplicação

As fronteiras de contagem separa o projeto ou aplicação que está sendo contado de aplicações externas. São utilizadas para estabelecer o escopo do produto que está sendo medido, bem como, a propriedade do dado e das funções, identificando se eles pertencem à aplicação que está sendo contada.

4.3.2.3 Funções do Tipo Dados

As funções do tipo dados representam a funcionalidade provida ao usuário através de dados internos ou externos à aplicação, sendo definidas como:

- Arquivos Lógicos Internos (ALI): segundo IFPUG (2000), consiste de um grupo lógico de dados relacionados, identificável pelo usuário, ou informação de controle[2] mantidas dentro da fronteira do aplicativo. Os quais, conforme Aguiar (2000), são mantidos através de entradas externas (apresentadas na seção 4.3.2.4).

---

[2] Informação de controle é o dado usado pela aplicação para garantir a aderência da aplicação aos requerimentos das funções do negócio estabelecidas pelo usuário. Por exemplo, em um sistema de recursos humanos um usuário pode especificar os controles necessários para a geração de relatórios sobre os funcionários da empresa.

- Arquivos de Interface Externa (AIE): a definição do IFPUG (2000), é um grupo de dados relacionados, identificável pelo usuário, ou informações de controle, referenciados pelo aplicativo, porém mantidos dentro da fronteira de um outro aplicativo. Isto significa que um AIE contado para um aplicativo deve ser um ALI para outro aplicativo.

A contagem das funções de dados possui duas regras básicas:

- Regras de Identificação
  - Identificação de Arquivos Lógicos Internos
  - Identificação de Arquivos de Interface Externa
- Regras de Complexidade
  - Identificação dos Registros Lógicos
  - Identificação dos Itens de Dados

Regras de Identificação

Segundo Braga (1996), se faz necessário identificar no que se refere aos Arquivos Lógicos Internos:

- Grupo de Dados ou Informação de Controle identificados pelo usuário como sendo um requisito da aplicação;
- Grupo de Dados mantidos através de um processo elementar dentro da fronteira da aplicação.

No que se refere aos Arquivos de Interface Externa, deve-se identificar (Braga, 1996):

- Grupo de Dados ou Informação de Controle identificados pelo usuário como sendo um requisito da aplicação;
- Grupo de Dados externos referenciados pela aplicação sendo contada;
- Grupo de Dados **não mantidos** pela aplicação que está sendo contada;
- Grupo de Dados mantidos em um Arquivo Lógico Interno de outra aplicação.

Sendo que a documentação a ser utilizada constitui-se em:

- DER (Diagrama de Entidade e Relacionamento)
- Listagem de Tabelas do DE-R, contendo os atributos das tabelas (campos das tabelas - nível lógico)

Regras de Complexidade

Existem regras bem definidas pelo IFPUG (2000), e que também podem ser encontradas em Braga (1996) e Hazan (2000), que determinam que elementos devem ser contabilizados como Registros Lógicos Referenciados (RLR) e Dados Elementares Referenciados (DER). A partir da contagem destes registros/dados, utiliza-se as tabelas referenciadas abaixo para se obter o valor de ponto por função bruto.

A complexidade funcional dos Arquivos Lógicos Internos é identificada de acordo com o Quadro 3.

Quadro 3: Identificação da Complexidade de Arquivos Lógicos Interno

	Itens de dados referenciados		
Número de Registros Lógicos	De 1 a 19	De 20 a 50	51 ou mais
Apenas 1	simples	simples	média
De 2 a 5	simples	média	complexa
6 ou mais	média	complexa	complexa

Fonte: Braga (1996)

Os Arquivos Lógicos Internos contribuem para a contagem de Pontos por Função não ajustados, baseando-se em sua complexidade funcional. Para traduzir Arquivos Lógicos Internos em Pontos por Função não Ajustados, utiliza-se o Quadro 4.

Quadro 4: Contagem de Pontos por Função dos Arquivos Lógicos Internos

Simples	**Médio**	**Complexo**
7 PF	10 PF	15 PF

Fonte: Braga (1996)

A complexidade funcional dos Arquivos de Interface Externa é identificada de acordo com o Quadro 5.

Quadro 5: Identificação da Complexidade Arquivos de Interface Externa

	Itens de dados referenciados		
Número de Registros Lógicos	De 1 a 19	De 20 a 50	51 ou mais
Apenas 1	simples	simples	média
De 2 a 5	simples	média	complexa
6 ou mais	média	complexa	complexa

Fonte: Braga (1996)

Os Arquivos de Interface Externa contribuem para a contagem de Pontos por Função não ajustados, baseando-se em sua complexidade funcional. Utiliza-se o Quadro 6 para traduzir Arquivos de Interface Externa em Pontos por Função não Ajustados.

Quadro 6: Contagem de Pontos por Função dos Arquivos de Interface Externa

Simples	**Médio**	**Complexo**
5 PF	7 PF	10 PF

Fonte: Braga (1996)

4.3.2.4 Funções do Tipo Transação

Funções do tipo transação representam a funcionalidade provida ao usuário pelo processamento de dados em uma aplicação, sendo definidas como:

- Entradas Externas (EE): a definição detalhada, fornecida por Aguiar (2000), baseada no IFPUG considera EE como um processo elementar[3] no qual dados atravessam a fronteira de fora para dentro. Tais dados podem vir de uma tela de entrada de dados, por via telefônica ou através de outro aplicativo. Os dados podem ser informações de controle ou informações do negócio. No caso dos dados serem informações do negócio, serão utilizados para manter um ou mais arquivos lógicos internos. Se os

[3] Processo elementar é a menor atividade com significado para os negócios do usuário. Por exemplo, a definição de empregado feita pelo usuário final inclui salário e informações de dependentes. A menor unidade de atividade seria a de incluir um novo empregado.

dados forem informações de controle, não será necessário que atualizem um arquivo lógico interno.

- Saídas Externas (SE): IFPUG (2000) define uma SE como “um processo elementar que gera dados ou informações de controle, enviados para fora da fronteira do aplicativo”. Aguiar (2000), complementa que os dados criam relatórios ou arquivos de saída, que são enviados a outros aplicativos, e que, estes relatórios e arquivos são criados a partir de um ou mais arquivos lógicos internos e/ou arquivos de interface externa.
- Consultas Externas (CE): Aguiar (2000), detalhou a definição do IFPUG considerando CE como um processo elementar com componentes de entrada e saída, que resulta na recuperação de dados de um ou mais arquivos lógicos internos e/ou arquivos de interface externa. A informação recuperada e enviada para fora da fronteira do aplicativo. O processo de entrada não atualiza nenhum ALI e o lado de saída não contém dados derivados.

O cálculo da complexidade das Funções transacionais na Contagem de Pontos por Função Não Ajustados é baseado no número de Arquivos Referenciados e Itens de Dados. Bem como as funções do tipo dado, estes também possuem regras bem definidas para se determinar que elementos considerar para o cálculo dos pontos por função, podendo ser encontradas em IFPUG (2000), Braga (1996) e Hazan (2000).

Utiliza-se a matriz de complexidade, Quadro 7, para identificar a complexidade das Entradas Externas identificadas.

Quadro 7: Identificação da Complexidade da Entrada Externa

	Itens de dados referenciados		
Número de Arquivos Referenciados	De 1 a 4	De 5 a 15	16 ou mais
0 ou 1	simples	simples	média
2	simples	média	complexa
3 ou mais	média	complexa	complexa

Fonte: Braga (1996)

As Entradas Externas contribuem para a contagem de Pontos por Função não ajustados, baseando-se em sua complexidade funcional , conforme Quadro 8.

Quadro 8: Contagem de Pontos por Função da Entrada Externa

Simples	**Médio**	**Complexo**
3 PF	4 PF	6 PF

Fonte: Braga (1996)

Para identificar a complexidade das Saídas Externas identificadas, utiliza-se o Quadro 9.

Quadro 9: Identificação da Complexidade da Saída Externa

	Itens de dados referenciados		
Número de Arquivos Referenciados	De 1 a 5	De 6 a 19	20 ou mais
0 ou 1	simples	simples	média
De 2 a 3	simples	média	complexa
4 ou mais	média	complexa	complexa

Fonte: Braga (1996)

As Saídas Externas contribuem para a contagem de Pontos por Função não ajustados, baseando-se em sua complexidade funcional, conforme Quadro 10.

Quadro 10: Contagem de Pontos por Função da Saída Externa

Simples	**Médio**	**Complexo**
4 PF	5 PF	7 PF

Fonte: Braga (1996)

A matriz de complexidade, demonstrado no Quadro 11, deve ser utilizada para identificar a complexidade das Consultas Externas identificadas.

Quadro 11: Identificação da Complexidade da Consulta Externa

	Itens de dados referenciados		
Número de Arquivos Referenciados	De 1 a 5	De 6 a 19	20 ou mais
0 ou 1	simples	simples	média
De 2 a 3	simples	média	complexa
4 ou mais	média	complexa	complexa

Fonte: Braga (1996)

As Consultas Externas contribuem para a contagem de Pontos por Função não ajustados, baseando-se em sua complexidade funcional, conforme relação apresentada pelo Quadro 12.

Quadro 12: Contagem de Pontos por Função da Consulta Externa

Simples	**Médio**	**Complexo**
3 PF	4 PF	6 PF

Fonte: Braga (1996)

4.3.2.5 Determinação dos Pontos de Função

Uma vez levantada a definição de cada função e sua complexidade relativa, o Quadro 13 é utilizado para calcular os pontos por função não ajustados (brutos), podendo ser encontrada em IFPUG (2000), Braga (1996) e Hazan (2000).

Quadro 13: Tabela para Cálculo dos Pontos por Função Brutos

Tipo de Função	**Complexidade Funcional**	**Total por Complexidade**	**Total Tipo Função**
Arquivo lógico interno	nº funções simples x 7 = nº funções média x 10 = nº funções complexa x 15 =		
Arquivo de interface externa	nº funções simples x 5 = nº funções média x 7 = nº funções complexa x 10 =		
Entrada externa	nº funções simples x 3 = nº funções média x 4 = nº funções complexa x 6 =		
Saída externa	nº funções simples x 4 = nº funções média x 5 = nº funções complexa x 7 =		
Consulta externa	nº funções simples x 3 = nº funções média x 4 = nº funções complexa x 6 =		
Total de Pontos de Função Não Ajustados			

Fonte: Hazan (2000)

O fator de ajuste é baseado nas 14 Características Gerais dos Sistemas (CGS), listadas em Pressman (1995) e visualizadas no Quadro 14, que avaliam a funcionalidade geral da aplicação que está sendo contada. O fator de ajuste influencia os Pontos por Função não Ajustados em +/- 35% para obter-se os Pontos por Função Ajustados.

Quadro 14: Questões para Definição do Fator de Ajuste

1. O sistema requer *backup* e recuperação confiáveis?
2. São exigidas comunicações de dados?
3. Há funções de processamento distribuídas?
4. O desempenho é crítico?
5. O sistema funcionará num ambiente operacional existente, intensivamente utilizado?
6. O sistema requer entrada de dados *on-line*?
7. A entrada de dados *on-line* exige que a transação de entrada seja elaborada em múltiplas telas ou operações?
8. Os arquivos-mestres são atualizados *on-line*?
9. A entrada, saída, arquivos ou consultas são complexos?
10. O processo interno é complexo?
11. O código foi projetado de forma a ser reusável?
12. A conversão e a instalação estão incluídas no projeto?
13. O sistema é projetado para múltiplas instalações em diferentes organizações?
14. A aplicação é projetada de forma a facilitar mudanças e o uso pelo usuário?

Fonte: Pressman (1995)

Para se determinar o Valor do Fator de Ajuste, deve-se:

1) Avaliar cada uma das 14 características gerais do sistema em uma escala de zero a cinco (utilizando-se o Quadro 15) para determinar o Nível de Influência (NI);

Quadro 15: Escala de Influência

Grau	**Descrição**
0	Nenhuma influência
1	Influência mínima
2	Influência moderada
3	Influência média
4	Influência significante
5	Influência forte

Fonte: Hazan (2000)

2) Somar os níveis de influência de cada uma das 14 características gerais dos sistemas para obter-se o Nível de Influência Total (NIT).;

3) Utilizar a seguinte fórmula:

Fator de Ajuste = (NIT*0.01)+0.65

As fórmulas para completar o último passo do procedimento de Contagem de Pontos por Função, inclui fórmulas relativas aos três tipos de contagens, conforme (Braga, 1996):

- **Contagem de Pontos por Função de Projetos de Desenvolvimento (PFD)**: Este tipo de contagem é utilizado em projetos de desenvolvimento que pode envolver, ou não, conversão de cadastros já existentes. Sendo que o cálculo consiste de três componentes da funcionalidade:
    - Funcionalidade da aplicação (PFB): consiste das funções obtidas depois da instalação do software.
    - Funcionalidade de conversão (PFC): consiste das funções providas para converter dados ou necessidades específicas de conversão manifestadas pelo usuário.
    - Funcionalidade de ajuste da aplicação (FA): consiste no valor obtido conforme descrição anterior.

    Fórmula: PFD = (PFB + PFC) * FA

- **Contagem de Pontos por Função de Aplicações Instaladas (PFA)**: estabelece a Contagem Inicial de Pontos por Função para uma Aplicação. Esta contagem reflete as novas funcionalidades recebidas pelo usuário, após um Projeto de Desenvolvimento. A contagem de Pontos por Função da Aplicação não inclui requisitos de conversão.

    Fórmula: PFA = PF_NÃO_AJUSTADO * FATOR_AJUSTE

- **Contagem de Pontos por Função de Projetos de Manutenção (PFM)**: este tipo de contagem pressupõe um sistema já desenvolvido, cujo objetivo é quantificar um projeto de manutenção.

    Fórmula: PFM = [(INC + ALT + PFC) * FAD] + (EXC * FAA)

    Onde:

INC: Pontos de função brutos que foram incluídos na aplicação pelo projeto de manutenção. Refere-se as funções que foram adicionadas à aplicação.

ALT: Pontos de função que foram alterados na aplicação pelo projeto de manutenção. Refere-se as funções que sofreram alteração.

PFC: São os pontos de função que foram adicionados pelo processo de conversão.

FAD: Fator de ajuste da aplicação depois do projeto de manutenção/

EXC: Pontos de função brutos que foram excluídos da aplicação pelo projeto de manutenção. Refletem as funções que foram excluídas da aplicação.

FAA: Fator de ajuste da aplicação antes do projeto de manutenção.

4.4 Use Case Points

Use case points (UCP), segundo Ribu (2001), é um método de estimativa e dimensionamento de software baseado na contagem de casos de uso (*use case*). Reed (2001) descreve que o processo de estimativa da complexidade dos projetos baseado em casos de uso envolve quatro fatores distintos: atores, casos de uso, fatores técnicos e de ambiente.

4.4.1 Classificando atores e casos de uso

Para Ribu (2001) os UCP podem ser obtidos a partir da análise dos casos de usos do sistema. Sendo o primeiro passo a classificação dos atores envolvidos como simples, médio ou complexo, estes atores são caracterizados de acordo com o Quadro 16:

Quadro 16: Classificação dos Atores

Tipo do Ator	Descrição	Fator	Exemplo
Simples	Sistemas externos	1	A interface para o sistema de contabilidade ou para o sistema de cartão de crédito. Esses tipos de atores têm uma interface bem definida e são bastante previsíveis quanto às suas reações, à saída proporcionada pelo sistema em questão ou à entrada que ele recebe da interface.
Médio	Hardware ou temporizadores	2	Um temporizador necessário para disparar certos relatórios.

			Embora esses atores sejam previsíveis, eles requerem mais esforço para controla-los e usualmente são mais propensos a erros.
Complexo	Humanos	3	Um funcionário do Atendimento ao Cliente, um Fonecedor ou um funcionário da Contabilidade. Esses tipos de atores são os mais difíceis de controlar e são totalmente imprevisíveis.

O total dos pesos não ajustados do ator (*Unajusted actor weights* – UAW) é calculado considerando quantos atores de cada tipo e multiplicando cada total pelo fator correspondente.

Cada caso de uso é então definido como simples, médio ou complexo, dependendo do número de transações na descrição do caso de uso, incluindo cenários secundários. O Quadro 17 é usado para a contagem dos UCP.

Quadro 17: Classificando Casos de Uso

Tipo de caso de uso	**Descrição**	**Fator**
Simples	3 caminhos ou menos	5
Médio	4 a 7 caminhos	10
Complexo	Mais de 7 caminhos	15

Fonte: Reed (2001)

Cada tipo de caso de uso é então multiplicado pelo fator associado compondo o total dos pesos não ajustados dos casos de uso (*Unajusted use case weights* – UUCW).

O total de pontos de casos de uso não ajustados (*unadjusted use case points* - UUPC) é obtido através da fórmula:

$$UAW+UUCW=UUCP$$

4.4.2 Fatores técnicos e de ambiente

Neste método de estimativa são considerados os fatores técnicos do projeto, utilizando como base o Quadro 18 e atribuindo uma avaliação entre 0 e 5 para cada tópico. Zero significa que o fator é irrele vante e 5 significa que ele é essencial.

Quadro 18: Peso para os Fatores Técnicos

Fator técnico	**Peso**
Sistema distribuído	2
Tempo de resposta (desempenho)	1
Eficiência do usuário final	1
Complexidade do processamento interno	1
Código reutilizável	1
Fácil instalação	0,5
Fácil de usar	0,5
Portátil	2
Fácil de mudar	1
Concorrente	1
Características especiais de segurança	1
Acesso direto ao software	1
Necessidade de treinamento especial para o usuário	1

Fonte: Ribu (2001)

Após avaliar cada tópico, multiplica-se então o peso pela avaliação para obter o Total Fator Técnico. O Fator de Complexidade Técnica (*Technical Complexity* Factor - TCF) é calculado através da fórmula:

$$TCF = (0{,}6+(0{,}01*\textbf{Total Fator Técnico}))$$

O último fator a considerar trata dos níveis de experiência dos membros da equipe de projeto. Isto é chamado de fator ambiental (*Environment Factor* - EF). O Quadro 19 apresenta a distribuição dos pesos:

Quadro 19: Peso para os Fatores Ambientais

Fator ambiental	**Peso**
Usando um processo formal de desenvolvimento (metodologia)	1,5
Usuários têm experiência com algum aplicativo anterior	0,5
Experiência orientada a objeto	1
Capacidade do analista chefe	0,5
Motivação	1
Requisitos estáveis	2
Trabalhadores em tempo parcial	-1
Dificuldade de linguagem de programação	-1

Fonte: Ribu (2001)

Considerando o Quadro 19 deve ser atribuido um valor entre 0 e 5 para cada fator, observando o seguinte contexto:

- Para os quatro primeiros fatores, 0 significa nenhuma experiência no assunto, 3 significa experiência média e 5 significa especialista.

- Para o quinto fator, 0 significa nenhuma motivação para o projeto, 3 significa motivação média e 5 significa motivação alta.
- Para o sexto fator, 0 significa requisitos extremamente instáveis, 3 significa requisitos médios e 5 significa requisitos inalterados.
- Para o sétimo fator, 0 significa nenhum pessoal técnico em tempo parcial, 3 significa médio e 5 significa todo o pessoal técnico em tempo parcial.
- Para o oitavo fator, 0 significa linguagem de programação fácil de usar, 3 significa média e 5 significa linguagem de programação muito difícil de usar.

O Fator Ambiental (*Environmental Factor* - EF) é calculado através da fórmula:

$$EF = (1{,}4 + (-0{,}03 * \textbf{Total Fator Ambiental}))$$

4.4.3 Pontos de casos de uso

Com os três componentes já conhecidos: UUCP, TCF e EF, pode-se calcular o número final chamado de pontos de caso de uso (UCP – *Use Case Points*):

$$UCP = UUCP * TCF * EF$$

4.5 COCOMO (COnstrutive COst MOdel)

O modelo COCOMO (*Constructive Cost Model*) é um dos modelos mais amplamente aceito e aplicado para estimar custos e esforço, segundo Ghezzi et al. (1991) e Von Mayrhauser (1990). Royce (1998) acrescenta que o modelo COCOMO é um modelo popular, aberto e bem documentado.

Este modelo, desenvolvido por Boehm, considera que o tamanho é o fator principal para estimar o custo. Além do custo, este modelo também estima o esforço (em termos de pessoa-mês) necessário para desenvolver o projeto. Sendo que o esforço estimado inclui desenvolvimento, gerenciamento e tarefas de suporte, mas não inclui o custo de secretariado e outras pessoas que podem ser necessárias para a organização.

O COCOMO assume a seguinte hierarquia de modelos (Pressman, 1995):

- COCOMO básico é um modelo estático de valor simples que computa o esforço (e custo) de desenvolvimento de software como uma função do tamanho do programa expresso em linhas de código estimadas.
- COCOMO intermediário computa o esforço de desenvolvimento de software como uma função do tamanho do programa e de um conjunto de "direcionadores de custo" que incluem avaliações subjetivas do produto, do hardware, do pessoal e dos atributos do projeto.
- COCOMO avançado incorpora todas as características da versão intermediária, com uma avaliação do impacto dos direcionadores de custo sobre cada fase do processo de desenvolvimento.

Para ilustrar o COCOMO na seqüência é apresentada uma visão geral da versão básica. Para maior aprofundamento deve-se verificar a obra de Boehm (1981) cujo autor foi o desenvolvedor do modelo.

O COCOMO pode ser aplicado em três classes de projetos de software: (i) modo orgânico – projetos de software simples, relativamente pequenos, nos quais pequenas equipes com boa experiência em aplicações trabalham num conjunto de requisitos não tão rígidos; (ii) modo semidestacado – um projeto de software intermediário (em tamanho e complexidade) no qual equipes com nível de experiência misto devem atingir uma combinação de requisitos rígidos e não tão rígidos; (iii) modo embutido – um projeto de software que deve ser desenvolvido dentro de um conjunto rígido de restrições operacionais, de hardware e de software.

A equação abaixo indica como calcular o COCOMO básico:

$$E = a_b(KLOC)\exp(b_b)$$

$$D = c_b(E)\ \exp(d_b)$$

Onde E é o esforço aplicado em pessoas-mês, D é o tempo de desenvolvimento em meses cronológicos e KLOC, o número estimado de linhas de código do projeto (expresso em milhares). Os coeficientes a_b e c_b e os expoentes b_b e d_b são fornecidos no Quadro 20.

Quadro 20: COCOMO básico

Projeto de software	a_b	b_b	c_b	d_b
Orgânico	2.4	1.05	2.5	0.38
Semidestacado	3.0	1.12	2.5	0.35
Embutido	3.6	1.20	2.5	0.32

Fonte: Pressman (1995)

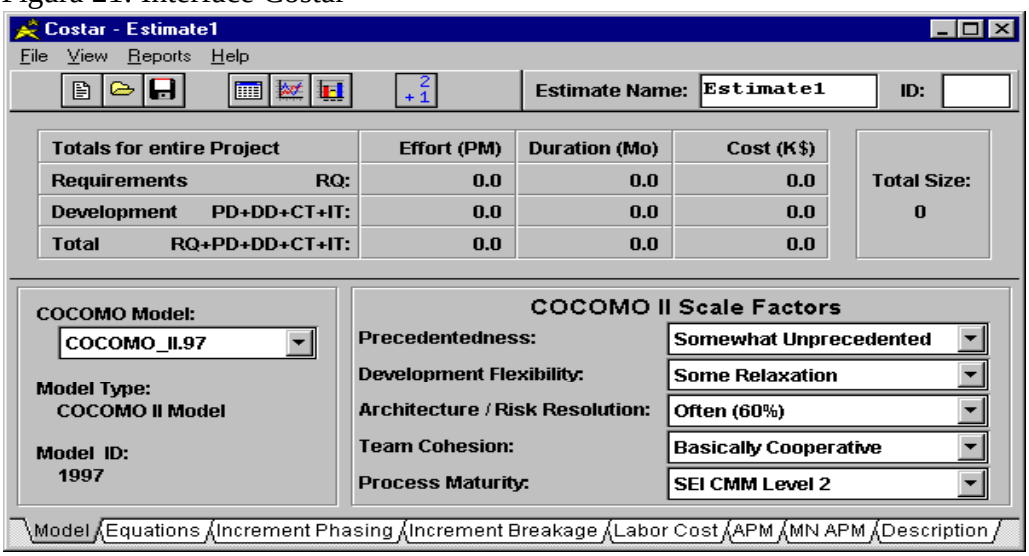
4.6 Ferramentas de Métricas de Software

Esta seção apresenta uma descrição sucinta de algumas ferramentas de métricas de software. Este estudo foi efetuado visando analisar as principais características das ferramentas existentes atualmente e, conseqüentemente, embasar a proposta apresentada. Vale ressaltar que esta lista refere-se apenas a um pequeno conjunto de ferramentas, existindo ainda outras disponíveis para se poder considerar uma análise exaustiva.

4.6.1 Costar

Costar é um software de gerenciamento de tempo, custos e avaliação de métricas para desenvolvimento de projetos, visualizado na Figura 21. Este software foi desenvolvido pela *Softstar Systems*, disponível em http://www.softstarsystems.com. A versão pesquisada foi desenvolvida em 1997 e, atualmente, encontra-se na versão 5.0, sendo que, para a análise foi utilizada uma versão demo cuja única limitação é o valor máximo permitido para o PF, tendo sido estabelecido um limite máximo de 500 pontos.

Figura 21: Interface Costar



A principal função do Costar é estimar o esforço (pessoas) e o tempo e custo de projetos, através da interligação de tarefas e pessoas. No que se refere ao esforço, são passados os dados de experiência, o cargo exercido, e o custo de determinada pessoa.

No que se refere a processo a ferramenta padroniza alguns modelos pré-definidos. Com base nos modelos apresentados o usuário pode alterar o início da iteração (etapa) seguinte, no entanto, não é permitido ao usuário configurar etapas diferenciadas.

No que tange ao esforço (pessoal) as informações requeridas (consideradas) são as experiências em máquina, linguagem, programas, plataformas, bem como, o horário de desenvolvimento, relação de tempo, dentre outras.

Quanto às informações pertinentes ao produto o software considera para o cálculo: tamanho de banco de dados, complexidade do produto, requerimento de recurso, dentre outros.

Para a geração das estimativas são consideradas todas as informações anteriormente citadas (exigências do desenvolvimento), tendo-se como resultado informações referentes a: esforços (em pessoas/mês), duração (meses) e custos (K$).

O software também fornece recursos para visualização gráfica, apresentando a distribuição de pessoal, marcos de referências e também alguns relatórios detalhados. Tais relatórios contextualizam, de maneira textual, informações referentes a esforço, o custo e a duração (tempo).

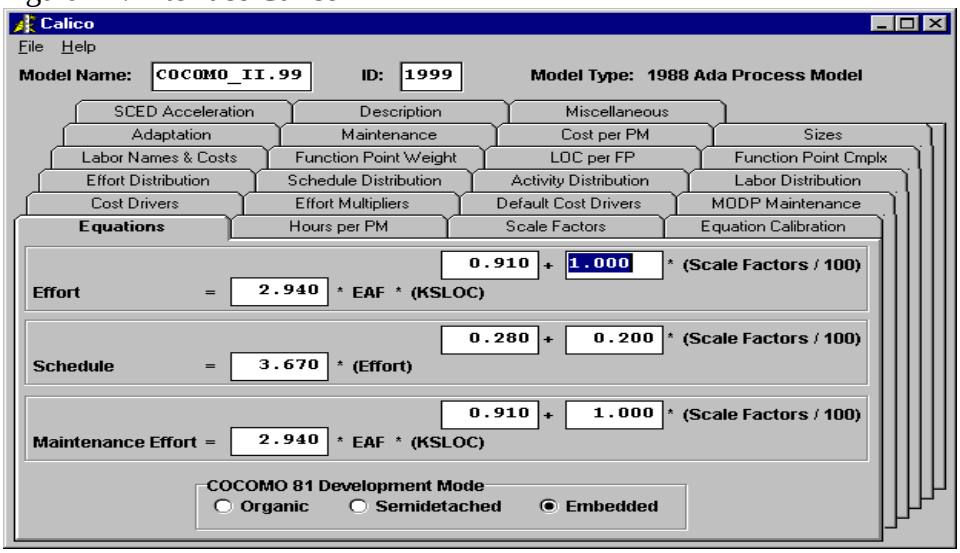
4.6.2 Calico

O Calico é um software que executa calibrações de equações do modelo COCOMO. Esta ferramenta foi desenvolvida pela empresa Softstar Systems (http://www.softstarsystems.com), sendo que, o Calico vem sendo utilizado para calibrar equações tanto no modelo COCOMO tradicional, como no modelo COCOMO II. A versão pesquisada foi desenvolvida em 1999 e, atualmente, encontra-se na versão 5.06, de modo que, para a análise foi utilizada uma versão demo cuja única limitação é o fato de não possuir um *Help on-line,* para guiar seus usuários. Baseado em dados referentes ao projeto (linguagem de programação utilizada, custo por mês dos

integrantes do projeto, horas de trabalho/mês, distribuição de esforço entre etapas, entre outros), o Calico calcula as equações ótimas para o projeto em questão.

O Calico também deixa modificar quaisquer variáveis que definem um modelo de COCOMO, pode-se editar os multiplicadores de esforço, mudar as equações (conforme visualizado na Figura 22), mudar as distribuições de esforço, entre outros.

Figura 22: Interface Calico



Informações referentes a técnica de Análise de Ponto de Função são transformadas para linhas de código, conversão esta viabilizada através da informação da linguagem de programação utilizada.

O software não fornece recursos para visualização gráfica, e nem relatórios textuais, todas as informações advindas desta ferramenta são apenas visualizadas na tela.

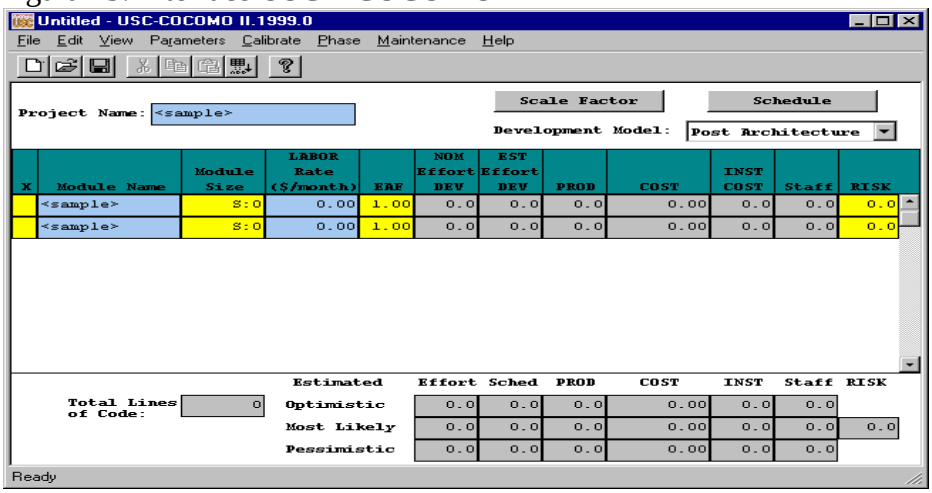
4.6.3 USC – COCOMO II

O software USC-COCOMO II foi desenvolvido pela *University of Southern* Califórnia. A versão estudada teve sua última atualização recentemente no ano de 1999. O USC-COCOMO II, ilustrado na Figura 23, é uma ferramenta CASE de análise que tem por fim gerar um resultado automatizado visando facilitar a administração de projetos de software.

USC-COCOMO estima o custo e tempo baseado em pessoa/mês e meses, respectivamente, para a determinação do *baseline* de exigências de um produto para a

conclusão de uma atividade. O software prevê um adicional de 20% ao tempo computado, como margem de erro (análise de risco).

Figura 23: Interface USC – COCOMO II



Para obter o custo por fase, dado que o COCOMO gera só o esforço/fase, é necessário gerar o custo médio por Pessoa-mês: calculado como custo total para projeto inteiro (excluindo exigências) / Esforço Total em Pessoa-meses (excluindo exigências). A este cálculo é assumido um adicional de 7% do Esforço Total.

O custo do projeto inteiro é a soma dos custos dos módulos individuais. Considerando que estas são só estimativas, são usadas tabelas para exibir o custo atual, gerando apenas uma estimativa otimista e pessimista para o custo do projeto inteiro.

Para definir o tamanho do programa torna-se necessário que se caracterize que medida será adotada (linhas de código, pontos por função ou adaptação). A adaptação é a reformulação ou adaptação em um programa pré-existente (as medidas por linhas de código e pontos por função foram abordadas nas seções 4.2 e 4.3).

O software implementa um método de calibração, aconselhado para menos de 8 projetos. Esta calibração realiza a análise dos dado que foram fornecidos ao programa e gera como resposta uma estimativa.

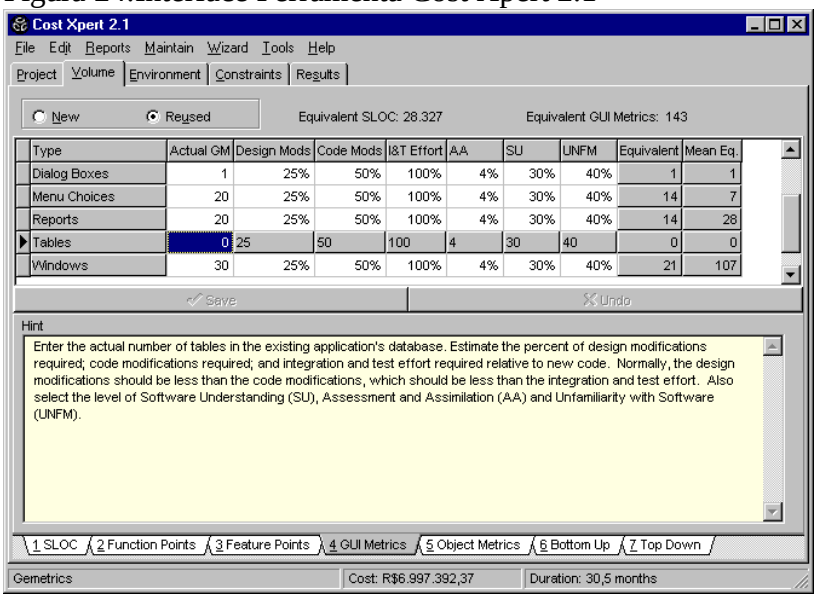
Através desta ferramenta CASE pode-se também gerar pequenos relatórios textuais com estimativas vinculadas a cada fase do processo. As fases correspondem ao ciclo de vida clássico (modelo cascata, seção 2.2.1) onde o esforço gasto em cada

atividade é computado como uma porcentagem do esforço total consumido durante uma fase.

4.6.4 Cost Xpert 2.1

Cost Xpert 2.1. é um software desenvolvido para estimar tempo e custo relacionado com o desenvolvimento de projetos de software. O software pesquisado tem sua última versão lançada pela empresa *Marotz Inc*. Podendo ser encontrado em http://www.marotz.com, atualmente na versão 2.1. Para análise foi utilizada uma versão demonstração com licença de uso para 45 dias sem limitações uso. A interface do software pode ser visualizada na Figura 24.

Figura 24:Interface Ferramenta Cost Xpert 2.1



A ferramenta mantém informações gerais sobre o projeto e usuário responsável pelos dados informados. No que se refere a dados relativos ao projeto, é possível enquadrar o projeto em diferentes tipos e associa-lo a padrões e ciclos de vida, sendo que o software fornece abertura para se configurar os parâmetros pré-definidos.

Para estimar o tamanho do projeto pode-se utilizar sete métodos para cálculo: SLOC, *Function Points*, *Feature Points*, GUI *Metrics*, *Object Métrics*, *Bottom Up* e *Top Down*.

Cost Xpert 2.1 possui compatibilidade com o MS-project, pois permite criar arquivos ".txt" e arquivos de dados de projeto que podem ser importados pelo MS-project e vice-versa.

O *software* realiza "análise de risco" calculando valores que afetam o esforço e também o tempo de entrega. Os fatores ambientais que podem afetar o custo global e duração de um projeto são divididos em categorias, tais como:

- Pessoal: considera fatores como experiência em análise, com o tipo de aplicação, linguagem e ferramenta, plataforma e programação.
- Plataforma: refere-se aos recursos para desenvolvimento.
- Projeto: considera ambiente e ferramentas disponíveis para desenvolvimento.
- Produto: refere-se a fatores relacionados com base de dados, documentação gerada durante o ciclo de vida, complexidade do produto, reusabilidade e tratamento a falhas.
- Fatores de escalonamento: considera fatores como riscos amenizados através do gerenciamento, flexibilidade dos requisitos, experiência de desenvolvimento do tipo de projeto, nível da empresa segundo CMM (*Capability Maturity Model*), interação da equipe (considerando equipe de desenvolvimento e também o *stakeholder*).

Cost Xpert 2.1 permite refinar a estimativa baseada em vários fatores. Entre os quais estão tempo *versus* custo, esforço requerido para planejamento, esforço para testar e integrar o projeto, tempo para revisão por parte do cliente e contatos com o cliente para solucionar dúvidas e fatores relacionados com problemas eventuais.

Os resultados podem ser visualizados através de relatórios e também graficamente.

4.7 Análise Comparativa das Ferramentas de Métrica de Software

Para a análise comparativa das ferramentas os critérios adotados foram:

- Linguagem por LOC: linguagens consideradas pela ferramenta para adaptação da quantidade de linhas de código, bem como se a ferramenta permite a inclusão de novas linguagens.
- Métricas/Indicadores: métricas ou indicadores implementados pela ferramenta.
- Processo: processo de desenvolvimento adotado pela ferramenta (caso exista).
- Visualização Gráfica: se a ferramenta gera algum gráfico comparativo para facilitar a análise dos resultados.
- Relatórios: forma de apresentação dos relatórios (caso emita).
- Recursos: recursos estimados por cada ferramenta.
- Help: se a ferramenta apresenta alguma forma de documentação.

A análise pode ser visualizada no Quadro 21.

Quadro 21: Análise Comparativa Software de Métricas

Critério/Ferramenta	Calico	Costar	Usc-Cocomo II	Cost Xpert 2.1
Linguagem por LOC	C, Fortran, Cobol, Basic, Pascal, Pl/1, Ada	C, Fortran, Cobol, Basic, Pascal, Pl/1, Ada	Vide lista apresentada abaixo (linguagens de conversão LOC*)	Inclui todas as demais, (exceto"*Report Generator*") possui mais de 500 incluindo variação de versão.
	Permite inclusão de nova Linguagem	Somente na definição de um novo modelo	Não permite inclusão de nova linguagem	Não permite inclusão de nova linguagem
Métricas/Indicadores	FP, COCOMO I , COCOMO II	FP, COCOMO_II.97, Early_Design, COCOMO_85, COCOMO_87, Ada,87, Apm_88	FP, Linhas de código, COCOMO II	FP, SLOC, *Feature Points, GUI Metrics, Object Metrics, Bottom Up, Top Down*
Processo	Não Contém	Processo iterativo (4 variações pré-estabelecidas)	Ciclo de vida Clássico (Cascata)	Não Contém
Visualização Gráfica	Não Contêm Visualização Gráfica	Contêm Visualização Gráfica	Não Contêm Visualização Gráfica	Contêm Visualização Gráfica

Relatórios	Não emite relatórios	Emite Relatórios com gráficos	Emite relatórios apenas textuais	Emite Relatórios com gráficos
Recursos	Esforço (pessoa-Mês), Duração (pessoa-Mês)	Esforço (pessoa-Mês), Custo, Duração (Mês) ,	Esforço (pessoa- Mês) Custo, Duração (Mês),	Pessoal, Custos, Duração (Meses)
Help	Inexistente	Documentação On-Line (F1)	Manual	Documentação On-Line (F1), sensível ao Contexto

**linguagens de conversão LOC implementadas pelo USC-COCOMOII: C, ANSI, Cobol 85, Forth, Fortran 77, Lips Modula 2, Pascal, Prolog, AI Shell, Fourth Generation, High - Level, Object Oriented, Program Generator, Query Language, Report Generator, Spreadsheet.*

4.8 Considerações Finais

Pelo estudo realizado em softwares para aplicação de métricas, pode-se observar que as ferramentas para métricas de software são poucas, e que possuem abrangência variável, ou seja, trabalham com conjunto de métricas diferentes, tendo-se como métricas amplamente aceitas os indicadores/métricas FPA, Linhas de código e algumas variações do COCOMO. A finalidade das ferramentas também é diversa podendo tanto ser para geração de estimativas quanto para calibrar a equação utilizada pelo modelo COCOMO.

Pode-se claramente observar com a revisão realizada que diversos autores ressaltam o uso de métricas de software como forma de fundamentar decisões de gerenciamento, e, no entanto, dentre as ferramentas analisadas, tanto de métricas quanto de gerenciamento, nenhuma delas efetua a integração de recursos de gerenciamento com apoio a métricas de software em uma mesma ferramenta. A partir deste fato, vislumbra-se a possibilidade de explorar esta lacuna como um diferencial entre as ferramentas apresentadas.

Sendo assim, como o objetivo final do trabalho é o desenvolvimento de uma metodologia suportada por uma ferramenta CASE, a seção seguinte apresenta brevemente conceitos e categorias referentes a ferramentas computadorizadas de auxílio a Engenharia de Software (CASE).

5 FERRAMENTAS CASE

A engenharia de software abrange um conjunto de três elementos fundamentais - métodos, ferramentas e procedimentos – que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade produtivamente, segundo Pressman (1995).

Os métodos de engenharia de software proporcionam os detalhes de "como fazer" para construir o software, e envolvem um amplo conjunto de tarefas que incluem: planejamento e estimativa de projeto, análise de requisitos de software e de sistemas, projeto, codificação, teste e manutenção.

As ferramentas de engenharia de software proporcionam apoio automatizado ou semi-automatizado aos métodos. Sendo denominadas de **Ferramentas CASE** (*Computer-Aided Software Engineering*).

Os procedimentos da engenharia de software constituem o elo de ligação que mantém juntos os métodos e as ferramentas e possibilita o desenvolvimento racional e oportuno do software de computador.

5.1 Taxonomia de Ferramentas CASE

As ferramentas CASE podem ser classificadas por função, por seus papéis como instrumentos para os gerentes e para o pessoal técnico, pelo uso que elas têm nas várias etapas do processo de engenharia de software, pela arquitetura do ambiente (hardware ou software) que as suporta ou até mesmo pela origem ou custo. A seguir são apresentados os diferentes níveis de tecnologia CASE, segundo Sommerville (1996) e uma taxonomia, apresentada por Pressman (1995).

5.1.1 Nível de Tecnologia CASE

Sommerville (1996) identifica três diferentes níveis de tecnologia CASE, que são:

- **Tecnologia de suporte ao processo de desenvolvimento:** inclui suporte para atividades do processo tais como especificação de requisitos, projeto,

implementação, teste, dentre outros. Segundo o autor, existem várias CASE deste nível, conseqüentemente, são as ferramentas mais "maduras".

- **Tecnologia de gerenciamento de processo:** inclui ferramentas de suporte a modelos de processo de desenvolvimento e de gerenciamento. O autor ressalta, e torna-se importante considerar pois se trata da classe de ferramenta proposta neste trabalho, que "existem poucos produtos disponíveis nesta área, mas ainda é assunto de consideráveis pesquisas".
- **Tecnologia Meta-CASE:** são ferramentas geradoras as quais são usadas para criar ferramentas de suporte a processo de desenvolvimento e ao processo de gerenciamento. O autor comenta que algumas ferramentas Meta-CASE estão disponíveis mas não são fáceis de usar e não tem sido largamente adotadas.

5.1.2 Classificação por Função

Pressman (1995) classifica as ferramentas CASE usando a função como critério primordial. Desta forma, as CASE são classificadas em:

- **Ferramentas de planejamento de sistemas comerciais:** ao modelar os requisitos de informação estratégicos de uma organização, as ferramentas constituem um "meta-modelo" a partir do qual sistemas de informação específicos são derivados. Em vez de se concentrar nos requisitos de uma aplicação específica, a informação comercial é modelada à medida que ela flui entre as várias entidade organizacionais dentro de uma empresa.
- **Ferramentas de apoio:** a categoria de ferramentas de apoio abrange ferramentas de aplicação e de sistemas que complementam o processo de engenharia de software. Entre elas incluem-se as ferramentas de documentação, as ferramentas de rede e de software básico, ferramentas de garantia da qualidade, ferramentas de gerenciamento de bancos de dados e gerenciamento de configuração (que são também membros da categoria de ferramenta de estrutura *framework*).
- **Ferramentas de análise e projeto:** possibilitam que o engenheiro de software crie um modelo do sistema que será construído. O modelo contém uma representação do fluxo de controle e de dados, conteúdo de dados, representações de processo, especificações de controle e uma variedade de outras representações de

modelagem. Este grupo de ferramentas se subdivide ainda em ferramentas AS/SD (ferramentas que implementam o método de análise estruturada e projeto estruturado), ferramentas PRO/SIM (de prototipação e simulação), ferramentas de projeto e desenvolvimento de interfaces e núcleos de análise e projeto.

- **Ferramentas de programação:** nesta categoria estão os compiladores, editores e depuradores que se encontram à disposição para apoiar a maioria das linguagens de programação convencionais. Além disso, os ambientes de programação orientados a objeto, as linguagens de quarta geração, os geradores de aplicações e as linguagens de consulta a bancos de dados também se situam nesta categoria.

- **Ferramentas de integração e testes:** Em seu catálogo de ferramentas de teste de software, *a Software Quality Engineering* define as seguintes categorias de ferramentas de testes:

ü Aquisição de Dados: ferramentas que adquirem dados a serem usados durante os testes.

ü Medição Estática: ferramentas que analisam o código-fonte sem executar os casos de testes.

ü Medição Dinâmica: ferramentas que simulam a função do hardware ou outros equipamentos externos.

ü Gerenciamento de Teste: ferramentas que auxiliam no planejamento, desenvolvimento e controle dos testes.

ü Ferramentas Transfuncionais: ferramentas que cruzam as fronteiras das categorias citadas.

- **Ferramentas de prototipação:** a prototipação é um paradigma da engenharia de software amplamente usado e, como tal, qualquer ferramenta que suporte pode ser legitimamente chamada “ferramenta de prototipação”. Por essa razão, muitas ferramentas CASE também podem ser incluídas nesta categoria.

- **Ferramentas de manutenção:** as ferramentas CASE para manutenção de software voltam-se a uma atividade que atualmente absorve aproximadamente 70% dos esforços relacionados a software. A categorização das ferramentas de manutenção pode ser subdividida nas seguintes categorias:

ü Ferramentas de Engenharia Reversa para especificação – capta o código-fonte como entrada e gera modelos gráficos de análise e projeto estruturados, listas "onde-usado" e outras informações de projeto.

ü Ferramentas de Análise e Reestruturação de Código – analisa a sintaxe do programa, gera um gráfico do fluxo de controle e gera automaticamente um programa estruturado.

ü Ferramentas de reengenharia de Sistemas On-Line – usadas para modificar sistemas de bancos de dados *on-line*.

- **Ferramentas de estrutura:** são ferramentas de software que oferecem gerenciamento de bancos de dados, gerenciamento de configuração e capacidades de integração de ferramentas CASE.

O presente trabalho está inserido no contexto deste último tipo de ferramenta CASE:

- **Ferramentas de gerenciamento de projetos:** podem exercer um profundo impacto sobre a qualidade do gerenciamento de projetos para esforços de desenvolvimento de software tanto grandes como pequenos. Ao usar um conjunto selecionado de ferramentas CASE, o gerente de projeto pode gerar úteis estimativas de esforço, custo e duração de um projeto de software, definir uma estrutura de divisão de trabalho, viabilizar o planejamento de projeto e acompanhar projetos em base contínua. Além disso, o gerente pode usar a ferramenta para compilar métricas, que por fim oferecerão uma indicação da produtividade no desenvolvimento de software e da qualidade do produto.

Desta forma, o capítulo 3 apresenta aspectos relacionados com gerenciamento de projetos, e que foram citados por Pressman (1995), como pertinentes a Ferramentas de gerenciamento de projetos.

Outra forma de classificar as ferramentas CASE é utilizando o critério de fragmentação, apresentado por Martin & Odell (1995), categorizando as ferramentas em I-CASE, CASE fragmentada e IE-CASE.

As ferramentas CASE fragmentadas suportam apenas parte do ciclo de vida do software, como por exemplo ferramentas de *front-end* para análise e ferramentas *back-end* para geração de código.

O termo I-CASE refere-se a um conjunto de ferramentas em que as ferramentas para cada fase do ciclo de vida do desenvolvimento são integradas, utilizando um único repositório logicamente coerente. Conjuntos de ferramentas I-CASE completos tem atualmente ferramentas para planejamento, análise, projeto e construção de protótipos, completa geração de código e teste. A manutenção é realizada modificando-se os diagramas do projeto e gerando novamente o código.

As ferramentas IE-CASE suportam a engenharia de informação e ajudam seus usuários a construir um modelo da empresa e a analisar áreas de negócios.

6 AGENTES

O termo **agente** é utilizado na literatura computacional para determinar diversos tipos de programas. Trabalhos mais recentes, segundo Spector (*apud* Costa, 1999), enfatizam que um agente pode ser qualquer sistema autônomo que percebe e age para alcançar um estreito conjunto de metas dentro de um específico ambiente virtual ou real.

Alguns dos atributos mais relevantes para compor um agente são:

- Autonomia: é a capacidade do agente de executar o controle sobre suas próprias ações, segundo Franklin & Graesser (1996). Ou seja, sem a necessidade de serem guiados por humanos.
- Comunicabilidade: quando existe mais de um agente envolvido, ressalta-se a necessidade por um modelo de comunicação. Conforme Genesereth & Ketchpel (*apud* Souza, 1996), os agentes comunicam-se através de troca de mensagens em uma linguagem específica para conversação.
- Cooperação: pode ser entendida como a capacidade que os agentes tem de trabalharem em conjunto de forma a concluírem tarefas de interesse comum.
- Reatividade: é a propriedade que permite aos agentes perceberem seus ambientes e responderem adequadamente às mudanças neles ocorridas.
- Flexibilidade: habilidade dos agentes em escolher que ações e em que seqüência deve tomá-las em resposta a um evento do ambiente.(Auer, 1995)
- Aprendizagem: Belgrave (*apud* Souza, 1996) relaciona as propriedades de aprendizado e comportamento adaptativo e as define como a habilidade apresentada pelo agente de acumular conhecimento baseado em experiências anteriores, e conseqüentemente, modificar seu comportamento em resposta a novas situações.

- Mobilidade: é a capacidade de transportar-se de uma máquina para outra, conforme Franklin & Graesser (1996). Agentes com a capacidade de mover-se através de uma rede de computadores podem auxiliar seus usuários na busca de informações. (Costa, 1999)

Além dos atributos listados pode-se citar ainda: Comportamento Adaptativo, Confiabilidade, Degradação Gradual, Discurso, Habilidade Social, Inteligência, Persistência, Personalização, Planejamento, Pró-Atividade, Representabilidade e Responsabilidade como alguns atributos pertinentes a alguns tipos de agentes (o tópico tipologia é abordado na seção 6.1)

Dentre todos os atributos listados, os 5 primeiros (autonomia, comunicabilidade, cooperação, reatividade e flexibilidade) são as principais características encontradas nos agentes propostos para viabilizar o gerenciamento distribuído de projetos de software.

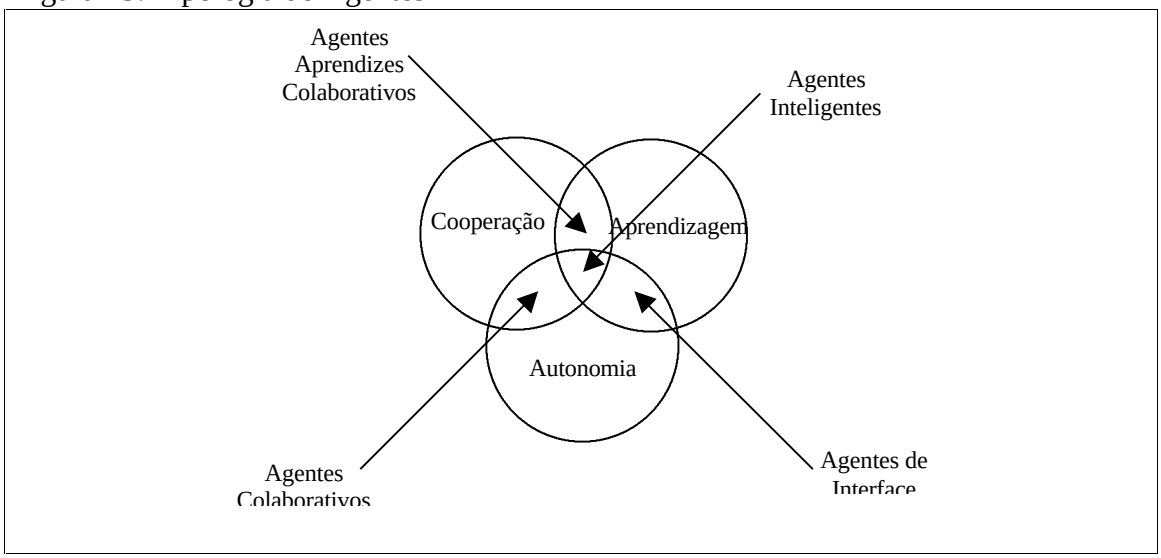
6.1 Tipologia de Agentes

O conjunto destes atributos é utilizado como uma forma para agrupar os agentes em classes ou tipologias. Devido a grande quantidade de atributos torna muito difícil a implementação de um agente que as englobe totalmente, até mesmo porque as características de um agente são dependentes do tipo de aplicação a que ele se propõe.

A análise dos atributos que estão presentes nos agentes tem sido utilizada pelos pesquisadores para organizar os agentes em tipologias. Uma tipologia é uma classificação por tipos de agentes que possuem atributos em comum.

Nwana (*apud* Costa, 1999) propõe uma tipologia de agentes com base nos atributos primários autonomia, cooperação e aprendizado. Combinando estas três características (Figura 25), quatro tipos de agentes podem ser derivados: agentes colaborativos com capacidade de aprendizado, agentes de interface, agentes inteligentes e agentes colaborativos.

Figura 25: Tipologia de Agentes



Fonte: Nwana (*apud* Costa, 1999)

A classificação citada anteriormente - agentes colaborativos - engloba o agente aqui proposto. Detalhando-se, portanto, este tipo, verifica-se conforme Costa (1999), que em sistemas colaborativos, cada agente contribui com sua própria técnica para a solução de um problema complexo. Agentes colaborativos enfatizam autonomia e cooperação com outros agentes de forma a executar tarefas para seus donos. Neste ambiente, torna-se clara a necessidade de negociação para estabelecer acordos e comprometimentos mútuos.

Para que um processo de colaboração possa acontecer, é clara a necessidade de se definir uma linguagem comum para a comunicação entre agentes, que constitui o principal tópico desenvolvido para viabilizar o presente trabalho.

Outras tipologias foram propostas para classificar os agentes, sendo estas vinculadas a: (i)quanto ao nível de inteligência, (ii)as tarefas que executam, (iii)a mobilidade, (iv)a aquisição de inteligência e, (v)a ênfase dada a alguns atributos primários.

6.1.1 Quanto ao Nível de Inteligência

Conforme River (*apud* Souza, 1996), quanto ao nível de inteligência podemos classificar os agentes em 3 níveis: baixo, médio e alto:

- Nível de Inteligência Baixo: Neste nível, os softwares agentes desempenham tarefas rotineiras, disparadas por eventos externos. Estes agentes executam redes de regras

complexas, não se adaptam a mudanças e não adquirem “esperteza” com o passar do tempo;

- Nível de Inteligência Médio: Neste nível, os softwares agentes utilizam uma base de conhecimento para desenvolver raciocínio em eventos monitorados. Podem adaptar-se a mudanças de condições na base de conhecimento e manipular as novas condições, porém normalmente não adquirem "esperteza” com o passar do tempo;
- Nível de Inteligência Alto: Neste nível, os softwares agentes utilizam tanto aprendizado quanto raciocínio na base de conhecimento. Aprendem com o comportamento do usuário, adquirem "esperteza" com o tempo e podem adaptar-se a mudanças de condições de ambiente.

6.1.2 Quanto a Tarefa que Executam

Segundo Jennings (1995), quanto a tarefa que executam podemos classificar os agentes como: Gopher, Prestadores de Serviço e Pró-Ativo:

- Gopher: Neste nível, os softwares agentes são considerados como agentes muito simples que executam tarefas diretamente, baseados em hipóteses, regras pré-estabelecidas e suposições. Por exemplo: o agente pode avisar o usuário que ele possui uma reunião marcada para as 14:00;
- Prestadores de Serviço: Neste nível, os softwares agentes executam tarefas de alto nível e bem definidas, quando requisitadas pelo usuário. Por exemplo: estes agentes podem organizar uma reunião (negociar datas e horários da reunião com os participantes);
- Pró-Ativo: Neste nível, os softwares agentes desempenham as tarefas mais complexas, eles podem pesquisar informações ou executar tarefas para o usuário sem serem requisitadas, sempre que isto for julgado apropriado. Por exemplo: um agente pode monitorar novos grupos sobre a Internet e retornar discussões que ele acredita serem de interesse do usuário.

6.1.3 Quanto a Mobilidade

Segundo Nwana (*apud* Giese, 1998), os agentes podem ser classificados de acordo com sua capacidade de se deslocar na rede através das seguintes formas:

- Agentes Estáticos: São Agentes que não podem mover-se através da rede, ou seja, eles ficam fixos em um único local;
- Agentes Móveis: São Agentes que possuem a habilidade de mover-se através da rede.

6.1.4 Quanto a Aquisição de Inteligência

Segundo Nwana (*apud* Giese, 1998), os agentes podem ser classificados quanto a Aquisição de Inteligência em deliberativos e reativos.

Sendo os agentes deliberativos, também chamados de simbólico ou cognitivo, contém uma representação explícita e um modelo simbólico do mundo, no qual decisões são tomadas por raciocínio lógico baseado em combinações de padrões e manipulação simbólica. O modelo é de certa forma pré-concebido, mas seu estado é alterado pelo agente em resposta a novas informações sobre o ambiente, percebidas pelos sensores do agente. O agente estima que ações serão necessárias para alcançar um determinado objetivo através da interpretação deste modelo, e então executa ações que levarão a sua realização.

Já os agentes reativos, que também podem ser chamados de reflexivos, ao contrário dos agentes deliberativos, não possuem um modelo simbólico interno de seus ambientes, atuando através de uma utilização de um tipo de comportamento estímulo/resposta, respondendo para o estado presente do ambiente no qual eles estão. São agentes simples que possuem um mapeamento de situações e respostas associadas. Assim, quando um determinado estado ambiental ocorre, o agente executa a ação associada.

6.1.5 Ênfase em Atributos Primários

Conforme Nwana (*apud* Giese, 1998), pode-se identificar vários tipos de agentes de acordo com seus atributos. Um agente não possui a necessidade de ter todas as propriedades.

- Agentes Colaborativos: São agentes que enfatizam autonomia e cooperação com os outros agentes para executar suas próprias tarefas;

- Agentes de Interface: São agentes que enfatizam autonomia e aprendizado para executar suas tarefas, interagindo com o usuário, recebendo especificações e entregando resultados;
- Agentes Móveis: São agentes escritos tipicamente em linguagem script, podendo ser enviado de um computador cliente para um computador servidor remoto para execução;
- Agentes de Informação: São agentes que acessam várias fontes de informações, e são capazes de colecionar e manipular informações obtidas destas fontes para responder consultas solicitadas pelo usuário ou outros agentes;
- Agentes Reativos: São agentes que executam tarefas quando solicitado pelo usuário, através do comportamento estímulo/resposta;
- Agentes Híbridos: São os agentes que combinam a filosofia de um ou mais tipos de agentes;
- Sistemas Heterogêneos: São um conjunto de dois ou mais agentes que pertencem a duas ou mais classes de agentes diferentes. Um sistema de agente heterogêneo pode também conter um ou mais agentes híbridos. Estes tipos de agentes também são entendidos como Sistemas Multi-Agentes.
- Agentes Autônomos: São agentes que podem interagir independentemente e efetivamente com seus ambientes, sem a participação do usuário.

6.2 Arquitetura de Agentes

Arquitetura de agentes é definida por Maes (*apud* Wooldridge & Jennings, 1995) como

> "uma metodologia particular para definir agentes. Especifica como o agente pode ser decomposto na construção de um ambiente de módulos componentes e como estes módulos podem interagir. O conjunto de módulos e suas interações devem prover uma resposta para a questão de como os sensores de dados e o estado interno corrente do agente determinam suas ações e futuro estado interno. Uma arquitetura deve prever as técnicas e algoritmos para suportar esta metodologia".

As arquiteturas podem ser divididas em três áreas: a arquitetura deliberativa (abordagem clássica do paradigma de IA), a arquitetura reativa (abordagem alternativa) e arquiteturas híbridas (utiliza ambas abordagens). A seguir são apresentadas algumas arquiteturas de agentes:

- Arquitetura de Quadro-negro: os sistemas de quadro-negro fornecem uma estrutura de dados central, denominado quadro-negro (blackboard), a qual é dividida em regiões ou níveis. Nesta arquitetura todas as interações ocorrem através do quadro-negro. Os agentes lêem e escrevem em um ou mais níveis sob a supervisão de um mecanismo global de escalonamento. (Souza, 1996)
- Arquitetura Baseada em Troca de Mensagens: os agentes comunicam-se entre si através da troca de mensagens e para tanto, torna-se necessário que os nomes dos agentes sejam conhecidos. A organização das interações é feita, com base em protocolos que definem as etapas da conversação entre os agentes para cada tipo de interação possível na sociedade. Os protocolos e os formalismos para representação de mensagem podem ser bastante variados.
- Arquiteturas Cognitivas: são associadas a agentes complexos, com mecanismos de inferência e decisão robustos, interações sofisticadas e alto grau de intencionalidade no comportamento. Esta arquitetura divide-se em:Arquiteturas Funcionais e Arquiteturas Baseadas em Estados Mentais.
- Arquiteturas Funcionais: o agente é dividido em módulos que implementam as funcionalidades consideradas necessárias a sua operação. Segundo Demazeu (*apud* Souza, 1996) a arquitetura do agente cognitivo possui: conhecimento, percepção, comunicação, decisão e raciocínio.
- Arquiteturas Baseadas em Estados Mentais: as arquiteturas baseadas em estados mentais adotam uma perspectiva psicológica para definição da estrutura de agentes. Os componentes mentais: crença, capacidades, escolha e compromisso, devem ser definidos de forma precisa e ter uma correspondência direta com seu uso no senso comum, conforme Oliveira (*apud* Souza, 1996).
- Arquiteturas Reativas: são próprias de agentes que possuem estrutura interna simples e interagem de forma limitada; geralmente não possuem representação dos estados

mentais; o desempenho da sociedade é resultado do número de agentes e da rapidez nas interações. Oliveira (*apud* Souza, 1996)

- Arquitetura de Suposição: é uma hierarquia de comportamentos realização-tarefas, onde cada comportamento compete com outros para exercer controle sobre um robô, conforme (Wooldridge & Jennings, 1995).
- Arquitetura Pengi: propõe que uma arquitetura eficiente de agente deve ser baseada na idéia de "execução de argumentos". Neste modelo foi desenvolvido o sistema PENGI, que pressupõe a maioria das tarefas como rotineiras. Assim, grande parte das decisões rotineiras podem ser codificadas em uma estrutura de baixo nível, que necessita apenas de atualização periódica para tratar novos tipos de problemas. (Souza, 1996)

7 METODOLOGIA E FERRAMENTA PARA GERENCIAMENTO DISTRIBUÍDO DE PROJETOS

A partir do apresentado no capítulo 2, se observa que a organização do processo de software é uma atividade crítica que engloba tanto o gerenciamento de diversos aspectos, tais como: tarefas, recursos e tempo, bem como, o gerenciamento de todos os produtos produzidos durante o ciclo de vida do sistema. A organização do processo também envolve a definição de métodos e ferramentas dentro de metodologias a serem aplicadas no desenvolvimento e gerenciamento do projeto.

De acordo com Carvalho & Chiossi (2001), os métodos são linhas gerais que governam a execução de alguma atividade e as ferramentas dão suporte à aplicação de métodos e metodologias.

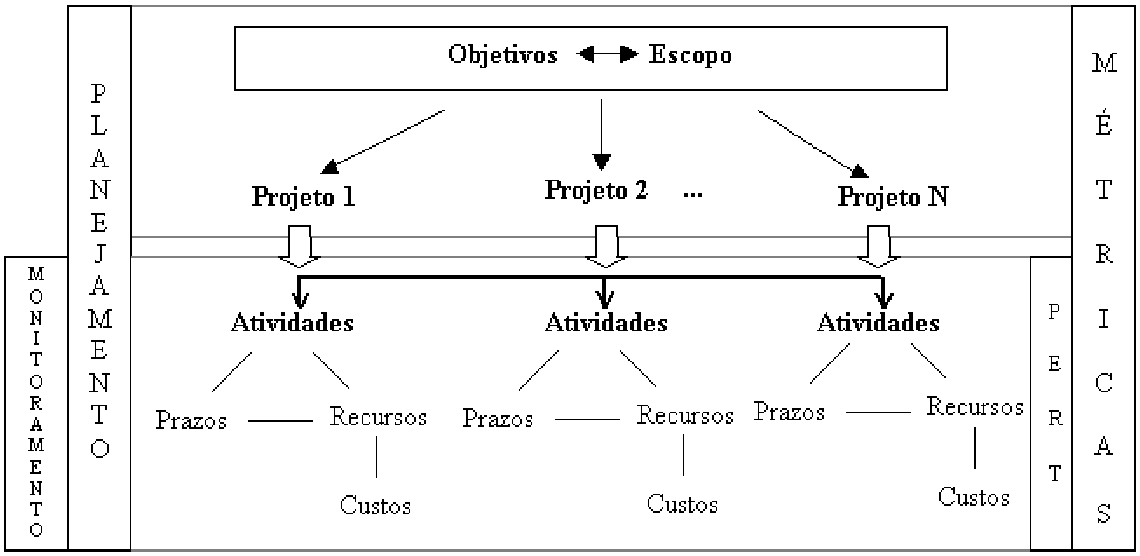
Conforme Carvalho & Chiossi (2001), "uma metodologia de desenvolvimento detalha as atividades do ciclo de vida, especificando um conjunto único e coerente de princípios, métodos, linguagem de representação, normas, procedimentos e documentação, que permitem ao desenvolvedor de software implementar sem ambigüidade as especificações advindas das fases do ciclo de vida do software".

Assim, este capítulo apresenta uma metodologia utilizando e/ou adaptando métodos já existentes na literatura para o gerenciamento distribuído de projetos, contemplando também a especificação de uma ferramenta CASE de apoio à metodologia. No capítulo 9 é apresentado o protótipo da ferramenta que suporta a metodologia e os métodos propostos, juntamente com um exemplo de aplicação demonstrando a viabilidade da metodologia e da ferramenta CASE.

7.1 Metodologia para Gerenciamento Distribuído de Projetos

A metodologia para gerenciamento distribuído trabalha com os mesmos elementos existentes no gerenciamento de qualquer trabalho, no entanto, prevê a interação entre diversos projetos e métodos para auxiliar no processo de desenvolvimento. Uma visão geral dos aspectos relacionados com a metodologia está retratada na Figura 26.

Figura 26: Visão Geral da Metodologia



A metodologia proposta divide o gerenciamento de projetos em 2 etapas: (i) planejamento e (ii) Monitoramento e Controle. Inicialmente, na etapa de planejamento, prevê a definição dos objetivos e do escopo do software a ser desenvolvido, para posteriormente, dividir a questão em planejamento em diversos projetos. Cada projeto conta com um conjunto de atividades que estão relacionadas entre si. Ainda cada projeto contém um conjunto de recursos e um custo atrelado a sua execução.

Após concluída a primeira etapa segue-se a etapa de monitoramento, tendo como objetivo principal acompanhar o planejamento efetuado garantindo o andamento do projeto dentro do prazo e custo estabelecido.

Para auxiliar no gerenciamento de projetos a metodologia prevê o uso de métrica de software, visando quantificação dos aspectos relacionados ao processo de obtenção de um produto com o objetivo de promover melhorias no processo de desenvolvimento e nos produtos, e visando otimizar o planejamento de prazos das atividades a metodologia adapta métodos tradicionais tais como PERT/CPM.

Desta forma, para o detalhamento da metodologia os seguintes itens são detalhados em cada etapa visando elucidar todos os aspectos envolvidos na metodologia:

- Objetivos: detalha os objetivos a serem alcançados através da utilização dos procedimentos focados na etapa.
- Terminologia: são conceituados os termos específicos da metodologia. Faz-se necessário para a correta compreensão dos procedimentos propostos.
- Premissas: descreve as condições assumidas como verdadeiras para a utilização da metodologia.
- Procedimentos: detalha passo-a-passo a seqüência de procedimentos considerados por esta metodologia para o gerenciamento distribuído de projetos.
- Métricas: descreve como se pode utilizar as métricas de software na etapa em questão.
- Métodos: apresenta os métodos para gerenciamento de projetos que são utilizados nesta metodologia e quais as adaptações necessárias.
- Resultados esperados: descreve os benefícios obtidos com o emprego desta metodologia.

7.1.1 Planejamento

Objetivo: Esta fase tem como objetivo identificar os requisitos do aplicativo. Entende-se como requisitos do projeto:

- Objetivos do projeto
- Escopo do software
- Atividades: refere-se ao que exatamente deverá ser desenvolvido.
- Custo aproximado
- Recursos utilizados: equipamentos, recursos humanos, dentre outros.
- Tempo para conclusão do projeto

Terminologia: Antes de descrever os procedimentos se faz importante definir o significado de alguns termos adotados nesta metodologia.

Aplicativo: produto gerado a partir do desenvolvimento de N projetos. Por exemplo, o aplicativo de gerenciamento de uma biblioteca pode ser obtido através do projeto do

módulo de consulta e catalogação e do projeto do módulo para realização de empréstimo, considerando que cada projeto tem um gerente próprio.

Atividade correlata: atividades planejadas nos projetos correlatos.

Link/Vínculo: ligação entre duas atividades que indica que há uma relação de precedência.

Projeto: parte que comporá um aplicativo. Cada projeto possui planejamento e gerenciamento próprio.

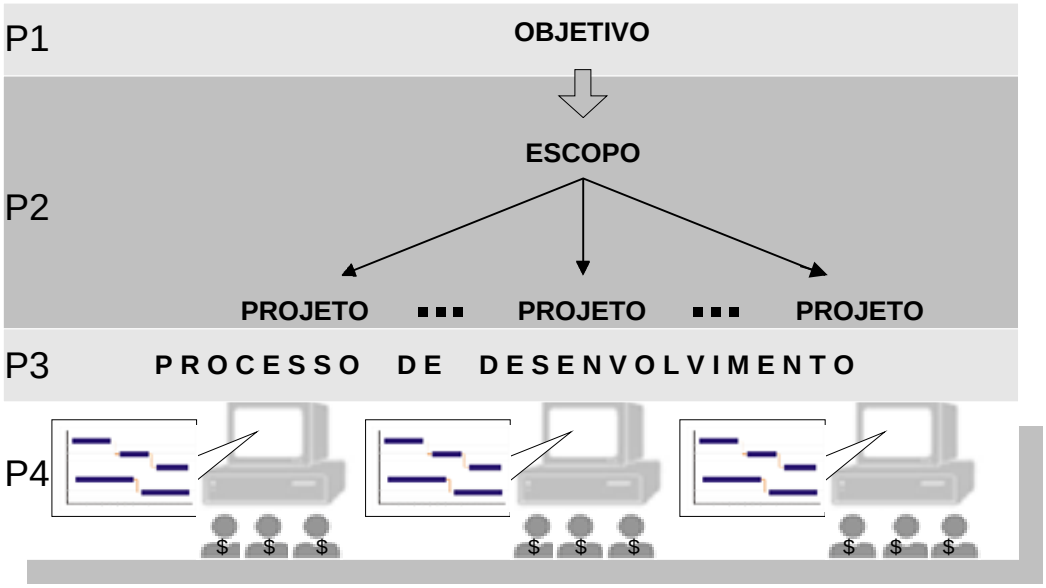
Projeto correlato/ Projeto distribuído: nome dado aos demais projetos que constituirão o aplicativo. Ou seja, considerando a visão de um dos gerentes, os demais projetos que darão origem ao aplicativo são correlatos ao seu.

Premissas: As seguintes premissas são consideradas verdadeiras para o desenvolvimento desta metodologia:

- A metodologia proposta não prevê uma hierarquia entre os gerentes de projetos, ou seja, não existe a figura de um coordenador geral, pois desta forma cada gerente é autônomo sobre as decisões de seu projeto. Cada gerente pode obter a visão global, no entanto, não pode fazer alterações nas atividades que não pertencem ao seu projeto.
- No mínimo deve-se estabelecer vínculos do tipo início-término entre as atividades. Ou seja, a atividade sucessora somente inicia após a finalização da atividade predecessora.

Procedimentos: Detalhando os procedimentos previstos para a etapa de planejamento, basicamente tem-se a seqüência apresentada na Figura 27 e apresentada abaixo.

Figura 27: Etapa de Planejamento Procedimentos 1 - 4



1) Definir o objetivo do aplicativo, os projetos envolvidos no desenvolvimento deste aplicativo e os gerentes responsáveis por cada projeto.

2) Definir o escopo do aplicativo e conseqüentemente o escopo de cada projeto relacionado.

3) Identificar o processo de desenvolvimento a ser adotado, definindo quais as fases envolvidas nos projetos. Este procedimento é importante, pois, viabiliza posteriormente uma análise do processo de desenvolvimento adotado auxiliando na maturidade da empresa.

Cada um dos procedimentos descritos a partir do item 4 deverão ser executados por cada gerente de projeto.

4) Detalhar as atividades a serem desenvolvidas para o cumprimento do projeto. Para cada atividade deve ser informado:

   a. A fase do processo de desenvolvimento a que pertence.

   b. Data de início prevista para a atividade.

   c. Tempo estimado para conclusão da atividade.

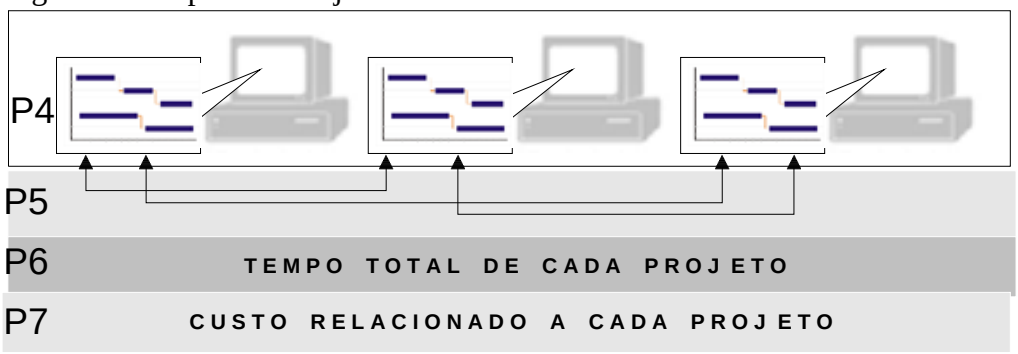
   d. Recursos alocados. Para cada recurso deve ser informado:

i. O tipo do recurso: equipamento ou humano. Caso seja alocado um recurso humano informar o cargo associado.

ii. Valores associados.

Uma vez determinadas as atividades pertinentes a cada projeto deve-se estabelecer os vínculos existentes entre os projetos relacionados (etapa 5). Estes vínculos são criados estabelecendo as relações de precedência entre as atividades dos diferentes projetos, conforme ilustrado na Figura 28.

Figura 28: Etapa de Planejamento Procedimentos 4 - 7



5) Estabelecer os vínculos entre as atividades. Estes vínculos podem ser criados tanto entre atividades de um mesmo projeto quanto com atividades de projetos correlatos.

6) Com o procedimento 5 concluído pode-se verificar o tempo associado ao projeto.

7) Através do procedimento 4 pode-se verificar os custos associados ao projeto. Este dado pode ser obtido através dos custos associados a cada recurso alocado e o tempo da atividade associada.

Métricas:

As métricas auxiliam em avaliar o próprio planejamento em relação a projetos já realizados. Através de valores quantitativos pode-se comparar entre projetos de mesmo tamanho os custos e tempo associado, permitindo um melhor planejamento e aperfeiçoamento do processo através de uma re-estruturação das atividades e recursos alocados. Para tanto se faz necessárias informações tais como:

- Estimar o tamanho do projeto, utilizando métricas tais como LOC, PF ou UCP.

- Histórico dos projetos desenvolvidos. Entende-se por histórico possuir planejamento detalhado de cada projeto desenvolvido anteriormente.
- Comparar o tempo de desenvolvimento entre projetos de tamanho semelhantes, caso haja.
- Comparar o custo entre projetos de tamanho semelhante, caso haja.
- Verificar o perfil profissional utilizado em projetos semelhantes, caso haja.

Métodos:

1) Recomenda-se utilizar a técnica de PERT/CPM para obter:

- Data de início mais cedo de uma atividade.
- Data de início mais tarde de uma atividade.
- Data de término mais cedo de uma atividade.
- Data de término mais tarde de uma atividade.
- Caminho crítico

Independente do método utilizado este deve considerar, para estimar as datas associadas às atividades e o tempo total do projeto, os vínculos criados entre atividades de projetos correlatos. Para isto é essencial a troca de informações entre todos os projetos correlatos.

2) Utilização de uma métrica de software para determinar o tamanho do projeto. Indica-se Pontos de função ou Linhas de Código.

Resultados Esperados:

Os principais resultados esperados desta fase são:

- Previsão do tempo para realização do projeto.
- Custos envolvidos no projeto.
- Cronograma de desenvolvimento.
- Pontos críticos associados. Refere-se as atividades que encontram-se no caminho crítico e também aquelas atividades que possuem vínculos com atividades de outros projetos. Estas atividades devem ter um acompanhamento mais rigoroso do gerente, pois podem comprometer o desenvolvimento da aplicação.

Com as datas fornecidas pelo PERT/CPM o gerente pode entre outras ações:

- Identificar atrasos na conclusão das atividades que comprometem a data de entrega do projeto.
- Otimizar as datas previstas para abreviar o tempo de desenvolvimento do projeto.

A quantificação dos aspectos relacionados ao processo de desenvolvimento, assim como do produto, é importante pelas seguintes razões:

- No caso do processo de desenvolvimento, as medições podem permitir melhorias no processo, aumentando a sua produtividade;
- No caso do produto, as medições podem proporcionar informações a respeito de sua qualidade.

7.1.2 Monitoramento e Controle

Objetivo: Monitorar o andamento do desenvolvimento do aplicativo tendo como base o planejamento efetuado na primeira fase.

Terminologia: Antes de descrever os procedimentos referentes à fase de monitoramento se faz importante definir o significado de alguns termos adotados.

Visão Local: considera apenas as atividades planejadas para o projeto em questão, desconsiderando as atividades que compõem os projetos correlatos.

Visão Global: considera o planejamento de desenvolvimento do aplicativo, apresentando o planejamento de todos os projetos relacionados.

Premissas: Para considerar uma visão global da realidade do andamento do aplicativo todas as informações das atividades correlatas ao projeto devem estar atualizadas.

Procedimentos:

Cada gerente responsável por um projeto deve:

1) Monitorar o andamento do projeto, acompanhando e controlando as atividades planejadas em uma visão local. Esta visão fornece ao gerente o planejamento considerando apenas as atividades referentes ao seu projeto.
2) Monitorar o andamento do aplicativo. Para isto o gerente deve se utilizar de uma visão global do planejamento. A qual permite controlar todas as atividades correlatas.

3) Manter atualizado o percentual de conclusão das atividades planejadas. Esta etapa é de fundamental importância para que os projetos correlatos tenham uma visão global realística do andamento do aplicativo.

4) Em caso de alteração no planejamento inicial, deve-se:

    a. Manter consistente as respectivas datas de início e fim das atividades visando obter uma visão global atualizada. Ou seja, ao alterar o período planejado de determinada atividade, esta alteração deve ser informada a todos os projetos correlatos.

    b. Mudanças efetuadas nas atividades consideradas críticas, e, nos pontos de interação entre projetos, devem ser informadas aos gerentes responsáveis pelos projetos correlatos.

    c. Recalcular as datas fornecidas pela técnica de PERT/CPM.

Métricas:

Associando-se atividades a valores fornecidos por métricas de software pode-se saber quantitativamente o andamento do projeto

Métodos:

Para esta fase se faz necessário uma forma de visualização gráfica do *schedule* do projeto e rastrear o progresso das atividades. Sugere-se o uso do gráfico de Gantt ou do gráfico de PERT, que são ferramentas amplamente utilizadas para o monitoramento de projetos. No entanto, estas ferramentas da forma como são encontradas atualmente não apresentam uma notação para representar vínculos entre atividades de diferentes projetos e mapear o que nesta metodologia denomina-se uma visão global do projeto apresentando uma visão relativa a cada projeto.

Para isto sugere-se as seguintes alterações nas ferramentas citadas para se adaptarem a metodologia proposta, sendo que tanto o gráfico de Gantt quanto o PERT devem poder apresentar diferentes notações para representar:

a) Atividade pertencente ao projeto: representa todas as atividades em uma visão local.

b) Atividade pertencente a projetos correlatos, ou seja, uma representação para atividades correlatas.

c) Vínculo entre atividades de um mesmo projeto.

d) Vínculo entre atividades de projetos correlatos.

e) Percentual concluído da atividade.

Resultados Esperados:

Os principais resultados esperados desta fase são:

- Com o acompanhamento contínuo do projeto e com uma visão global do andamento do desenvolvimento o gerente possa ajustar seu planejamento inicial assim que os atrasos forem detectados. Pois, é consenso que quanto antes forem detectados os atrasos no cronograma maiores as possibilidades de se reverte o atraso e entregar o projeto no prazo proposto.
- Com o comprometimento de cada gerente em manter os dados de seu projeto atualizados os problemas são logo detectados o que compromete menos o andamento do projeto, e viabiliza uma melhor análise de risco.
- Há uma maior integração e comprometimento da equipe de trabalho, pois se tem troca de informações contínuas e conseqüentemente fornece uma visão realista das implicações de cada alteração de cronograma para os demais projetos correlacionados.

7.1.3 Considerações Finais da Metodologia

Considerando as 3 dimensões do gerenciamento de projetos – recurso, tempo e tarefa - esta metodologia prioriza no gerenciamento distribuído a dimensão tempo, pois é o principal fator considerado crítico no gerenciamento de projetos. E este fator torna-se ainda mais difícil de ser gerenciado quando se tem um planejamento e gerenciamento distribuído dos projetos.

Tal afirmação é evidenciada quando analisados os dados obtidos de um levantamento junto a 21 empresas da empresas da região de Florianópolis, Itajaí, Balneário Camboriú e Blumenau. Este levantamento utilizou o questionário que consta do Anexo 1. Quando consideradas as respostas as questões 5.3 “Qual dos fatores é

considerado pela empresa como fator crítico?" e da questão 5.6 "A empresa desenvolve ou já desenvolveu projetos em conjunto com outra empresa/instituição?" tem-se os resultados apresentados na Tabela 1.

Tabela 1: Fator Crítico *versus* Desenvolvimento Distribuído

Fator Crítico	Desenvolvimento distribuído	
	Sim	Não
Recurso	6	5
Tempo	8	5
Tarefa	2	1

Pode-se observar que o fator tempo é considerado crítico tanto pelas empresas que desenvolvem software considerando o desenvolvimento distribuído quanto pelas empresas que não desenvolvem desta forma. No entanto, o fator tempo é apontado com maior freqüência como crítico pelas empresas que trabalham com o planejamento/gerenciamento distribuído de software. Desta forma, o fator tempo foi priorizado pela metodologia proposta neste trabalho.

7.2 Especificação de Ferramenta CASE para Suporte a Metodologia

Esta seção apresenta a especificação dos principais requisitos para implementação de uma Ferramenta CASE de suporte a metodologia proposta. Para tanto, primeiramente é fornecida uma descrição dos requisitos a serem abordados e os métodos a serem implementados bem como a arquitetura proposta para o gerenciamento distribuído. A seção seguinte apresenta os diagramas de casos de uso fornecendo os cenários propostos.

7.2.1 Descrição Textual dos Requisitos

Inicialmente, visando uma melhor compreensão, os requisitos foram divididos em 3 grupos. Cada grupo contém a descrição detalhada de alguns requisitos e apresenta quais métodos/técnicas devem ser implementados, justificando as escolhas efetuadas. Assim, os requisitos foram estabelecidos com base nos seguintes grupos:

- Requisitos básicos: que contém os procedimentos gerais tanto de gerenciamento quanto das métricas a serem abordadas.
- Requisitos de apoio à decisão: compõem os relatórios a serem gerados para fornecer apoio a decisões gerenciais de projeto.

- Requisitos para gerenciamento distribuído: detalha os requisitos a serem controlados para viabilizar o gerenciamento distribuído de projetos.

7.2.1.1 Requisitos Básicos

A partir da metodologia proposta observa-se que a ferramenta deve ter como objetivos: (i) fornecer ao gerente de projeto informações relacionadas ao esforço, custo e duração de um projeto de software, (ii) definir uma estrutura de divisão de trabalho e (iii) viabilizar o planejamento e gerenciamento de projeto. Além disso, o gerente pode usar a ferramenta para compilar métricas, que por fim oferecerão uma indicação da produtividade no desenvolvimento de software e da qualidade do produto.

No que tange ao gerenciamento de projetos, observa-se que este freqüentemente acarreta várias questões conflitantes, tais como: não há tempo para executar a tarefa, o trabalho é muito complexo ou o orçamento não é adequado. Para proceder nestas situações, Strauss (1997) recomenda que se deve entender e considerar as 3 dimensões gerais do gerenciamento de projeto: tempo, tarefa e recursos. E são nestes fatores que é centrada a metodologia proposta e, conseqüentemente, o projeto da ferramenta, no que se refere a questão de planejamento.

Cada uma das dimensões será implementada, podendo-se defini-las e exemplificá-las da seguinte maneira:

- Tempo: o tempo requerido refere-se ao cronograma – especialmente ao deadline (data final). Esta data depende da natureza da tarefa (projeto) e da disponibilidade de recursos. Conforme indicado na metodologia, o algoritmo de PERT/CPM deve ser utilizado para apresentar informações relevantes (conforme apresentado na metodologia) referentes as datas do projeto.
- Tarefa: refere-se ao que exatamente está sendo desenvolvido. É o escopo do trabalho a ser realizado: a grandeza e a complexidade da aplicação final. Ou seja, consiste na especificação dos requisitos, no projeto funcional, dentre outros. Reflete-se no detalhamento das atividades a serem cumpridas no desenvolvimento do projeto.
- Recursos: basicamente se referem a quanto dinheiro está disponível para ser gasto no projeto e como o dinheiro é aplicado em termos de pessoas, material e equipamento.

Devem ser manipulados através da alocação de recursos vinculados as atividades planejadas.

A metodologia prevê uma forma de apresentação visual do planejamento do projeto, podendo para tanto se utilizar tanto o gráfico de Gantt quanto o gráfico de PERT. Segundo Sommerville (1992), uma das ferramentas mais familiares para visualizar o andamento de um projeto é o Gráfico de Gantt. Desta forma, optou-se pela implementação deste gráfico para o acompanhamento do projeto.

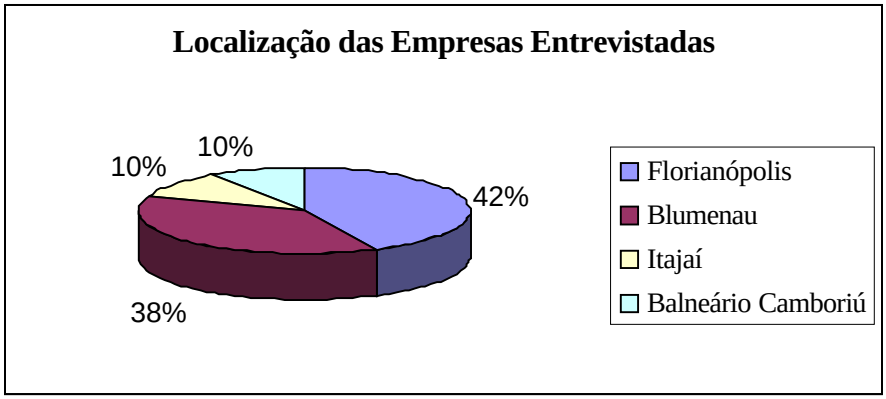
No que se refere a métricas de software, a metodologia prevê o emprego de uma métrica de software visando fornecer um melhor entendimento do processo utilizado para desenvolver um produto, assim como uma melhor avaliação do próprio produto.

Dentro deste contexto, a métrica a ser implementada pela ferramenta deverá ser pontos de função. Pois, segundo Aguiar (2001) presidente do Brazilian Function Point Users Group (BFPUG) e diretor do International Function Point Users Group (IFPUG):

> "os pontos de função são a única medida ao mesmo tempo independente de plataforma ou linguagem, compreensível pelo usuário e universal ... Trata-se de um padrão mundialmente reconhecido, que já existe há cerca de 15 anos ... Hoje, vemos diversas empresas trabalhando com pontos de função como forma de medir os resultados, o que provoca uma grande mudança no mercado. No governo, por exemplo, é extremamente comum a publicação de editais de licitação baseados em pontos de função."

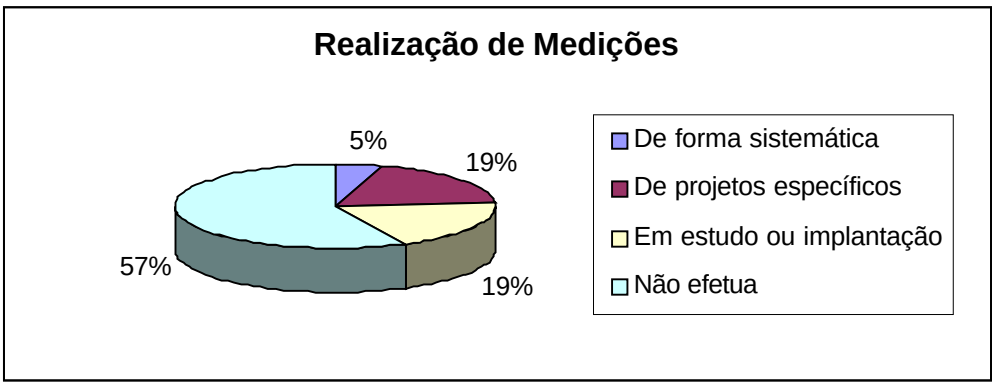
A escolha pela métrica pontos de função também foi embasada pelo levantamento efetuado através da aplicação de um questionário (Anexo 1) em 21 empresas de desenvolvimento de software das cidades de Itajaí, Balneário Camboriú, Blumenau e Florianópolis, distribuídas conforme gráfico da Figura 29. Para a aplicação do questionário utilizou-se uma amostragem aleatória simples, que considera que qualquer subconjunto da população tem a mesma probabilidade de fazer parte da amostra.

Figura 29: Gráfico Representando a Distribuição das Empresas Entrevistadas



Através deste questionário tem-se refletida a aplicação de métricas de software nas empresas entrevistadas, conforme Figura 30.

Figura 30: Gráfico com o Percentual de Empresas que Realiza Medições



Este gráfico demonstra que apesar da maioria das empresas não realizar medições, uma parcela significativa já efetua alguma forma de medição ou pretende implantar. E, dentre as empresas que utilizam alguma forma de medição, a métrica Pontos de Função é aplicada por todas as empresas, e apenas uma das empresas emprega, além dos pontos de função, a métrica COCOMO.

Também se considerou neste levantamento o tipo de software desenvolvidos (questão 4.1 do Anexo 1) pelas 21 empresas entrevistadas onde se observou que a distribuição do emprego de métrica PF por categoria ocorre conforme demonstrado na Tabela 2.

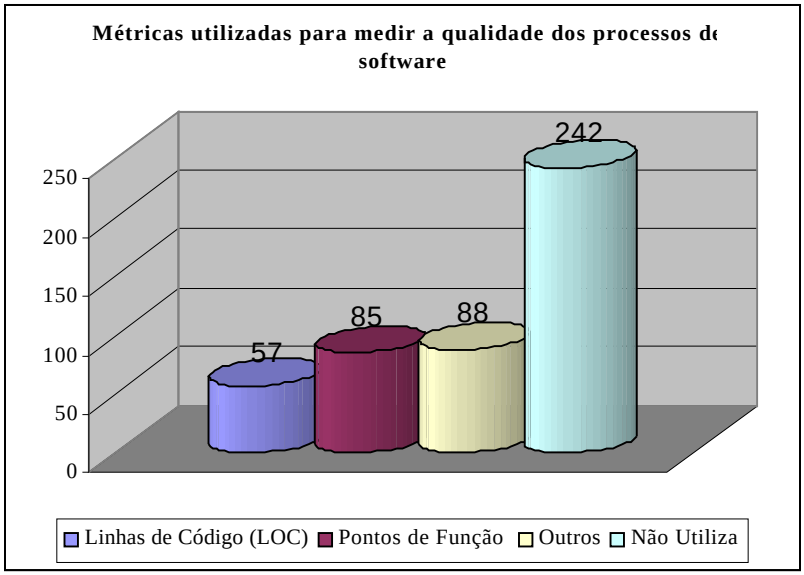
Tabela 2: Categoria de Software em que Foram Empregadas Métrica PF

Categorias	**Métrica Pontos por Função**	
	Qtd. empresas que utiliza	**Qtd. empresas que não utiliza**
Administração	3	5
Administração de recursos humanos	2	2
Automação bancária	1	0
Automação comercial	2	2
Automação de escritórios	2	1
Automação industrial	2	3
Banco de dados	2	3
Comunicação de dados	1	0
Construção civil	1	1
Contabilidade	1	2
Controle de qualidade de processo	1	1
Saúde	1	2
Transportes	1	0
Financeiro	3	4
Gerenciador de informações	1	2
Meio Ambiente	1	0

Pode-se notar neste levantamento que a métrica PF pode ser aplicada a diferentes tipos de software, sendo portanto, uma métrica bastante abrangente. As demais características referenciadas na questão 4.1 do Anexo 1 não constam da Tabela 2 por não o emprego de métrica relacionada as mesmas.

Um panorama nacional do uso de métricas considerando as empresas brasileiras pode ser encontrado em Qualidade e Produtividade no Setor de Software Brasileiro (2000), que lançado a cada 2 anos pela Secretaria de Política de Informática e Automação do Ministério da Ciência e Tecnologia apresenta o resultado de pesquisas diretas com empresas desenvolvedoras de software, visando acompanhar a evolução da gestão da qualidade neste setor. Verifica-se, na última versão lançada até o momento da realização deste projeto (referente ao ano 1999), que o panorama nacional encontra-se representado no gráfico da Figura 31.

Figura 31: Utilização de Métricas por Empresas Desenvolvedoras de Software



Fonte: Ministério da Ciência e Tecnologia (2000)

Conforme observado no gráfico da Figura 31, das empresas que utilizam alguma categoria de métrica para medir a qualidade do processo de software, a maioria (a questão permitia múltipla escolha) adota a métrica pontos de função. A pesquisa desenvolvida pelo Ministério da Ciência e Tecnologia envolveu 445 empresas desenvolvedoras de software, sendo 44 de Santa Catarina.

Assim, considera-se a métrica pontos de função a melhor opção a ser implementada, pois, além de atender os requisitos propostos na metodologia, agrega características viáveis de ampla utilização e uma boa aceitação pelas empresas tanto da região (como observado no questionário aplicado a 21 empresas da região) quanto pelas empresas nacionais (conforme relatado pelo Ministério da Ciência e Tecnologia (2000)).

7.2.1.2 Requisitos de apoio à decisão

Para obter maiores benefícios da integração entre aspectos de planejamento/gerenciamento de projetos e as métricas de software faz-se necessário gerar relatórios que visem apoiar a tomada de decisão por parte do gerente. Também se faz necessário implementar relatórios que fornecem informações referentes ao processo de desenvolvimento.

Assim, a ferramenta proposta deve gerar os seguintes relatórios para auxílio ao processo de desenvolvimento, ou seja, visa fornecer *feedback* sobre o processo de desenvolvimento utilizado, visando o seu aprimoramento:

- Custo por cargo: apresenta o custo médio (por mês) de cada cargo envolvido no projeto. É útil para calcular o custo total do projeto baseando-se na quantidade de horas de cada perfil (ou cargo) durante o desenvolvimento do projeto.
- Custo por fase: relaciona o custo (com recursos humanos e recursos físicos) envolvido em cada fase do ciclo de desenvolvimento, para uma aplicação específica.
- Alocação de recursos: apresenta a alocação, em horas de trabalho por dia, para cada recurso selecionado para atuar no projeto.
- Perfil profissional por fase: fornece a relação de horas trabalhas por cada cargo em cada fase do processo de desenvolvimento.
- Esforço por fase: relaciona cada pessoa envolvida nas diferentes fases do processo de desenvolvimento do projeto e o total de horas e percentual dedicado.

Além destes relatórios, os seguintes relatórios, com dados comparativos entre projetos, devem ser implementados:

- Custo do ponto de função entre projetos: demonstra a relação de valor para desenvolver um ponto de função em cada um dos projetos.
- Tempo de desenvolvimento de um ponto de função entre projetos: relaciona o tempo (em horas) para desenvolvimento de um ponto de função entre aplicações.
- Quantidade de pontos de função bruto entre projetos: demonstra como estão divididos os pontos de função brutos das aplicações. Estes números demonstram a composição média dos pontos de função de um sistema. Estas informação podem ser úteis quando necessita-se preparar uma estimativa do tamanho de um sistema em fase de levantamento em que se dispõe de poucas informações sobre o sistema.
- Comparativo entre fatores de ajustes entre projetos: apresenta o valor associado aos fatores de ajustes das diferentes aplicações cadastradas.
- Comparativo do tamanho dos sistemas planejados: emite um comparativo classificando os sistemas por tamanho em pontos de função.

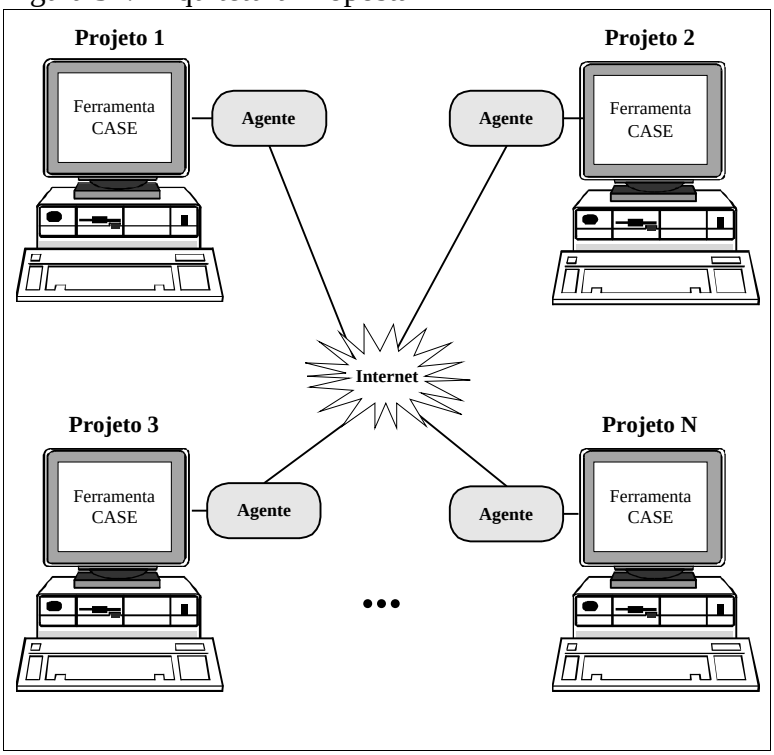
Os relatórios devem apresentar algum gráfico para facilitar a análise dos dados. Assim, todos estes relatórios buscam uma melhor qualificação de performance atual sobre três enfoques: produtividade, qualidade e custos. Ou seja, fornecendo informações essenciais para a gerência de projetos e ainda permitir a melhoria do processo de desenvolvimento e, conseqüentemente, do produto gerado.

7.2.1.3 Requisitos para Gerenciamento Distribuído

Esta seção apresenta uma visão simplificada da arquitetura proposta para realizar o planejamento e o gerenciamento distribuído de projetos, envolvendo basicamente 2 entidades – a ferramenta CASE e o agente– sendo, portanto, um sistema multi-agente, conforme visualizado na Figura 32.

Para compreender a natureza do gerenciamento distribuído pode-se considerar, ainda observando a Figura 32, que o produto a ser gerado consiste no resultado dos N projetos representados.

Figura 32: Arquitetura Proposta



Considera-se nesta arquitetura a ferramenta CASE como a interface através da qual se dá o planejamento e gerenciamento do projeto. Sendo que esta aplicação pode ser utilizada para o gerenciamento de qualquer tipo de projeto, podendo ser ou não distribuído. O agente é a entidade responsável pela consistência entre os projetos correlacionados. Portanto, é entidade indispensável para o planejamento e gerenciamento dos projetos distribuídos. A função desta entidade é monitorar as ações do usuário (gerente de projeto) sobre a ferramenta CASE, e, conforme as situações percebidas neste ambiente, o agente utiliza um conjunto de mensagens específicas (detalhadas na seção 8.3.2) para informar as alterações ocorridas aos demais agentes responsáveis pelos projetos correlacionados. Assim, se utiliza uma arquitetura baseada em troca de mensagens (conforme descrição na seção 6.2).

Detalhando-se, portanto, as funções dos agentes sobre o gerenciamento do projeto:

- Manter a coerência dos vínculos (*links*) entre atividades dos projetos correlacionados. Pois, em projetos distribuídos podem ocorrer casos em que a dependência de uma atividade está relacionada a uma atividade de outro projeto correlacionado. Neste caso, é função do agente manter consistente as respectivas datas de início e fim das atividades.
- Além da consistência entre as datas, os agentes devem manter os gerentes responsáveis avisados de mudanças efetuadas nas atividades consideradas críticas, pois são pontos de interação entre projetos. Exemplificando: se uma determinada atividade atrasa em um projeto x e, esta atividade é predecessora a uma outra no projeto y correlacionado, portanto, também provocará atraso no planejamento do projeto y e, este fato precisa ser do conhecimento do gerente responsável pelo projeto y. Atuar sobre estas situações e informá-las ao gerente é tarefa pertinente aos agentes.
- Atualizar a visão global do gerenciamento. A ferramenta permitirá construir o gráfico de Gantt sobre uma visão local ou total do projeto. Onde para a visão local apenas são listadas as atividades do projeto em questão e, para a visão total, o agente é ativado para atualizar a base de dados também com as atividades dos projetos correlacionados – exibindo todas as atividades em um único gráfico de Gantt.

- Todas as solicitações efetuadas/recebidas deverão ser mantidas em um arquivo de *log*. Desta forma, é possível garantir um gerenciamento efetivo, pois tem-se controle de solicitações atendidas e recusadas. As solicitações recusadas seriam mensagens enviadas e não recebidas pelo agente, devido a problemas de rede ou ao fato do agente não estar ativo. Neste caso, é implementado um ciclo de tempo no qual é rastreado o arquivo de *log* e as mensagens recusadas são re-enviadas, até a confirmação das mesmas.

A partir desta arquitetura proposta é possível observar os atributos, citados na no capítulo 6, pertinentes ao agente: autonomia, comunicabilidade, cooperação, reatividade e flexibilidade. Pode-se afirmar que a propriedade de autonomia está presente, pois, independente do gerente (usuário) estar utilizando a aplicação o agente permanece ativo, recebendo mensagens dos demais agentes e atualizando sua base de dados. E, ainda, dependendo das mensagens recebidas é função do agente comunicar ao gerente (usuário) mudanças que interfiram diretamente no seu planejamento (como atrasos em atividades críticas, por exemplo).

Já o atributo comunicabilidade é claramente percebido devido a existência de um conjunto de mensagens que viabiliza a troca de mensagens entre os agentes. Quanto a propriedade cooperação é notada quando observa-se as funções descritas como pertinentes aos agentes. Ou seja, a análise de riscos, bem como o gerenciamento efetivo dos projetos está fortemente atrelado a capacidade de cooperação entre os agentes.

O monitoramento efetuado pelo agente sobre a aplicação demonstra a característica de reatividade. Pois, é a partir de mudanças observadas no ambiente monitorado (no caso, a aplicação GEMETRICS) que o agente reage (normalmente através de envio de mensagens aos demais agentes). A situação inversa também ocorre, recebendo uma comunicação de alteração em algum projeto correlacionado o agente reage alterando dados na aplicação monitorada ou comunicando ao gerente (usuário) responsável.

A flexibilidade é observada no momento em que uma alteração é percebida e o agente autonomamente decide que operação efetuar (envio de mensagem, comunicação ao gerente, registro no arquivo de *log*, dentre outros).

Quanto ao nível de inteligência, mencionado por River (*apud* Souza, 1996), pode-se classificar o agente como possuindo baixo nível de inteligência pois, este agente executa tarefas rotineiras disparadas por eventos externos observados a partir do monitoramento da base de dados do aplicativo GEMETRICS. Seu comportamento é modelo a partir de regras, sendo que não adquire "esperteza" com o passar do tempo. Desta forma, Nwana (*apud* Giese, 1998), classifica o agente como sendo reativo pois seu comportamento é do tipo estímulo/resposta.

Considerando a classificação apresentada por Jennings (1995) o agente proposto é considerado um agente Pró-ativo, uma vez que realiza tarefas sem ser requisitado pelo usuário.

Uma característica importante a ser ressaltada trata-se do fato da arquitetura não necessitar de um servidor para permitir o gerenciamento distribuído. Desta forma, a ferramenta torna-se de fácil instalação e de simples utilização.

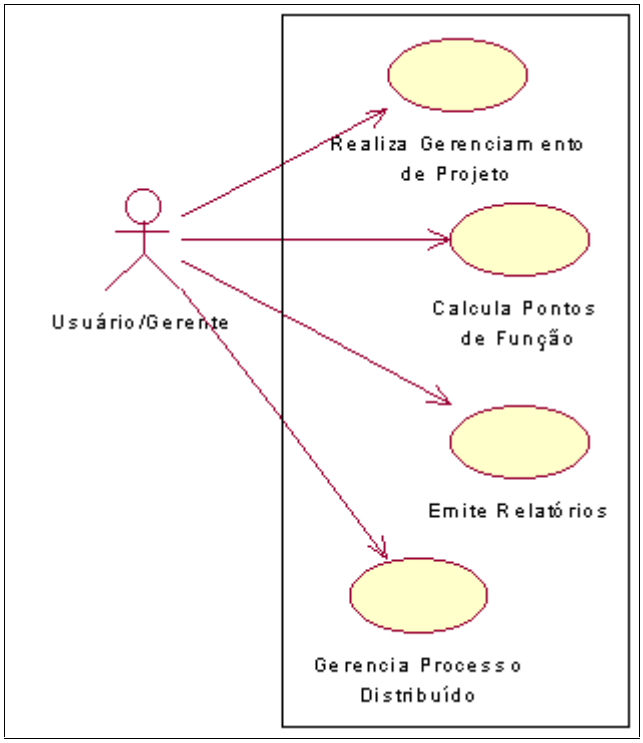
Uma visão possível desta arquitetura com ausência de um servidor (um centralizador das informações) é que as mudanças no ambiente podem não ser imediatamente processadas – no caso do agente estar inativo. No entanto, com a implementação de arquivos de *log* e ciclo de tempo pode-se garantir uma efetiva comunicação entre os agentes.

7.2.2 Diagramas de Caso de Uso

Para fornecer uma visão geral dos requisitos da ferramenta CASE GEMETRICS esta seção apresenta os diagramas de caso de uso. "Um diagrama de caso de uso mostra um conjunto de casos de uso e atores e seus relacionamentos" (Booch et al. 2000). Os autores afirmam que este diagrama é essencialmente importante para a organização e modelagem dos comportamentos de um sistema. Para elaborar os diagramas utilizou-se a Ferramenta CASE Rational Rose 2000© desenvolvida pela Rational Software Corporation.

Desta forma, a fim de elucidar, num primeiro momento, o contexto da ferramenta proposta, a Figura 33 apresenta o diagrama de caso de uso.

Figura 33: Modelagem do Contexto



Na seqüência, visando apresentar o comportamento a ser implementado na ferramenta são apresentados os principais requisitos da ferramenta CASE proposta. De forma que, para cada caso de uso apresentado na Figura 33, um novo diagrama de caso de uso foi desenvolvido detalhando o comportamento de cada domínio.

Portanto, conforme apresentado no levantamento dos requisitos e representado na modelagem do contexto os principais cenários envolvidos são:

- Realiza Gerenciamento de Projetos: trata do gerenciamento das tarefas (atividades), tempo de desenvolvimento e recursos. Além disso, apresenta o gráfico de Gantt como ferramenta de gerenciamento. O detalhamento deste cenário pode ser observado na Figura 34.
- Calcula Pontos de Função: este cenário abrange todos os aspectos relacionados com a métrica pontos de função. Incluindo cadastramento das funções tipo dado e transação, fatores de influência e dimensionamento do projeto. O detalhamento do comportamento deste cenário pode ser visualizado na Figura 35.
- Emite relatórios: engloba todos os relatórios para suporte a decisão, ressaltando aspectos de integração entre a métrica implementada e aspectos relevantes ao

gerenciamento de projeto. Os relatórios foram divididos em dois novos cenários: (i) aspectos referentes a um projeto específico, ilustrado no diagrama de casos de uso da Figura 36; (ii) relatórios comparativos entre projetos, retratados no diagrama de casos de uso da Figura 37.

- Gerencia processo distribuído: envolve o comportamento pertinente aos agentes incluídos na ferramenta para manter a consistência do planejamento e gerenciamento entre os projetos correlatos. A Figura 38 detalha os aspectos relevantes para a implementação dos agentes.

Figura 34: Diagrama de Casos de Uso -Realiza Gerenciamento de Projeto

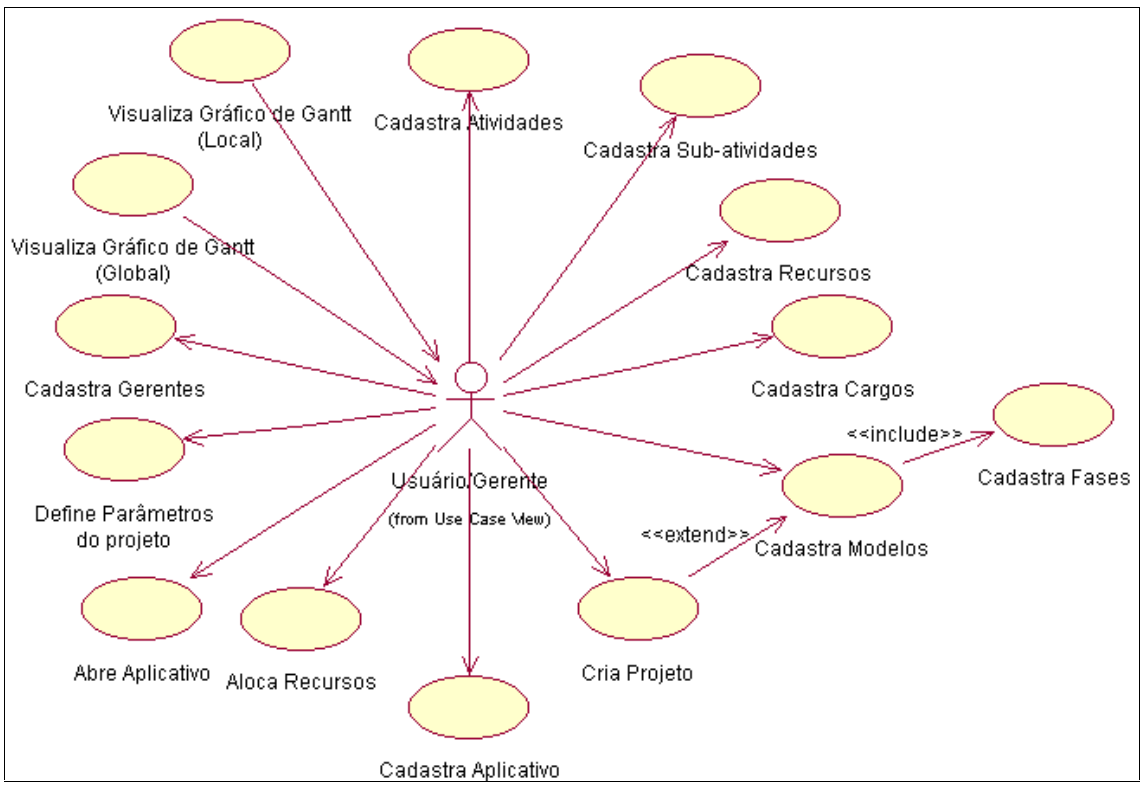
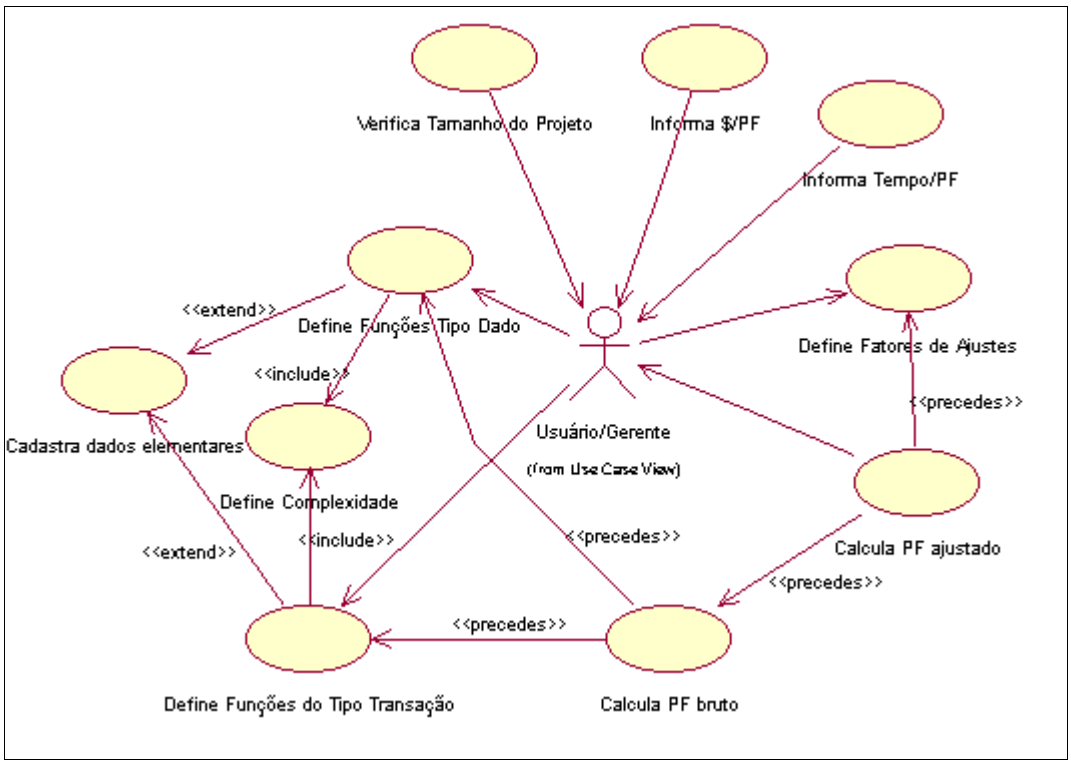


Figura 35: Diagrama de Casos de Uso - Calcula Pontos de Função



O diagrama de Caso de uso Calcula Pontos de Função (Figura 33) utiliza o estereótipo *precedes* que não é padrão da UML (*Unified Modeling Language*). Este estereótipo, no entanto, foi sugerido por Rosenberg (1999) onde o autor sugere seu uso para indicar que um caso de uso precisa preceder outro em uma seqüência lógica. Como este diagrama apresenta a forma de cálculo dos pontos de função e claramente existe uma seqüência lógica a ser seguida, optou-se por incorporar este estereótipo ao modelo a fim de torná-lo mais preciso.

Figura 36: Diagrama de Casos de Uso - Emite Relatório (projeto específico)

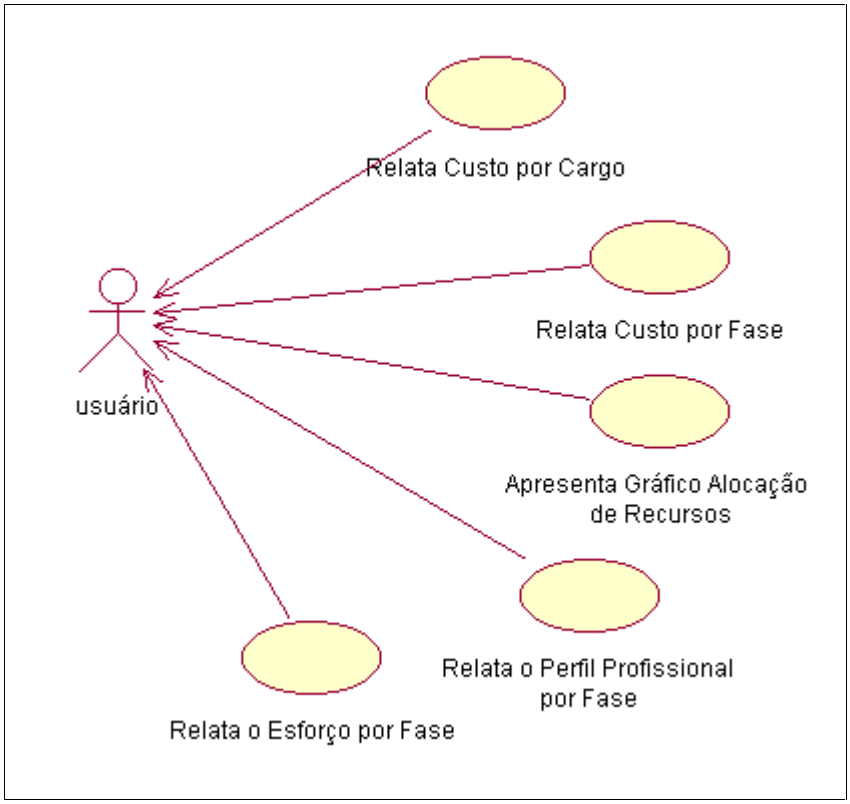


Figura 37: Diagrama de Casos de Uso - Emite Relatório

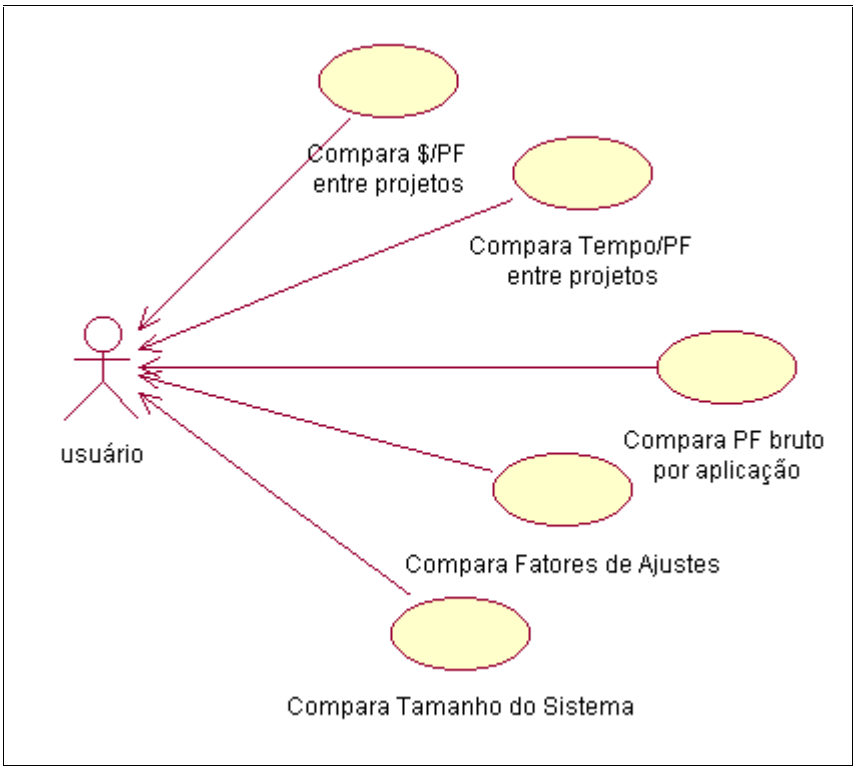
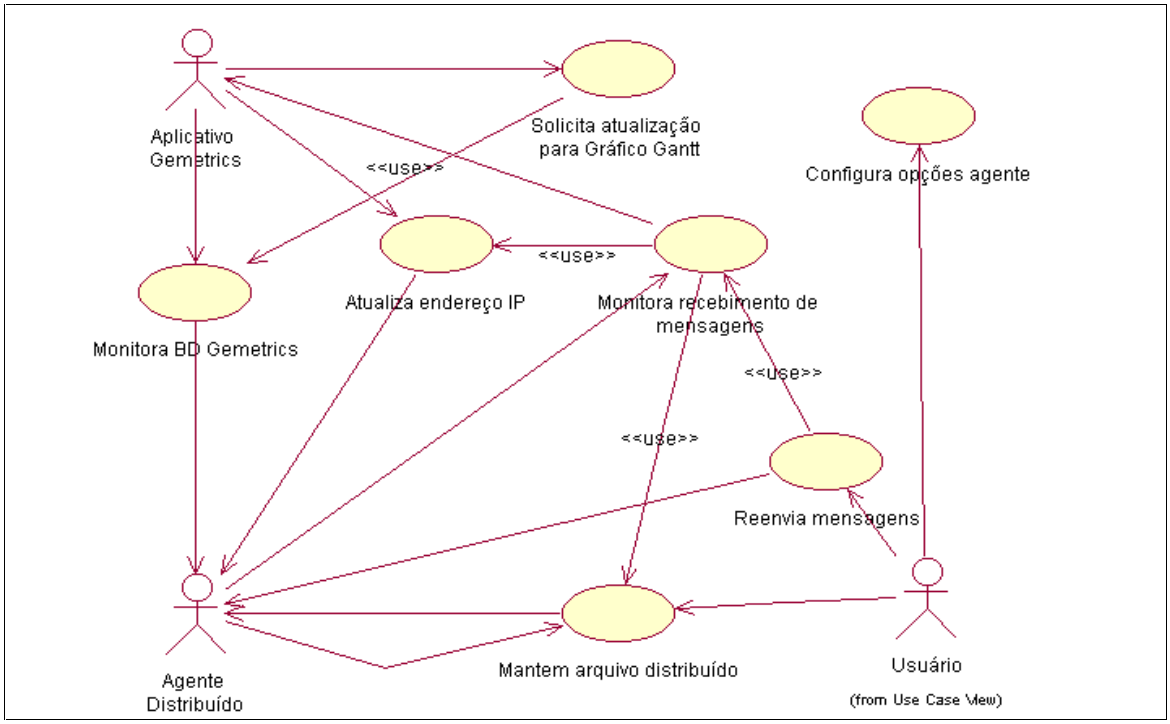


Figura 38: Diagrama de Casos de Uso - Agente para Gerenciamento Distribuído



O Apêndice A apresenta a especificação detalhada de cada caso de uso apresentado nos diagramas. A estrutura utilizada na especificação dos casos de uso apresenta-se como se segue:

- Nome do Caso de Uso: referencia o caso de uso especificado, utilizando o mesmo nome apresentado no diagrama.
- Categoria: informa o cenário ao qual pertence o caso de uso.
- Objetivo: apresenta resumidamente a função (objetivo) do caso de uso.
- Fluxo Principal: descreve a seqüência de eventos a serem realizados pelo caso de uso.
- Fluxo Alternativo: caso haja a possibilidade de uma seqüência de eventos diferente da descrita no fluxo principal, este item apresenta as possíveis variações.
- Fluxo de Exceção: descreve possíveis exceções ao funcionamento padrão descrito pelo fluxo principal.

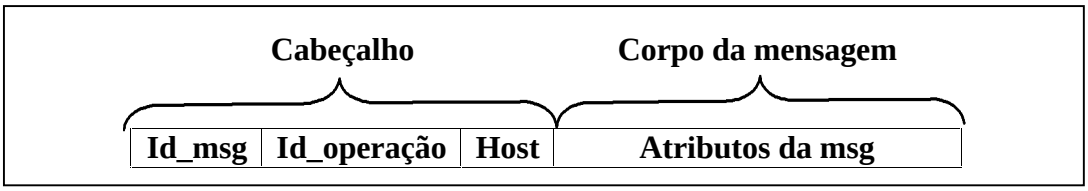
7.2.3 Formato Geral das Mensagens

Conforme especificado na seção 7.2.1.3, a arquitetura agente proposta consiste em uma arquitetura baseada na troca de mensagem. Assim, um conjunto de mensagens

para comunicação dos agentes se faz necessário. De maneira geral, as mensagens são compostas por um cabeçalho de identificação e o corpo da mensagem, conforme Figura 39. O cabeçalho contém:

- Id_msg: identificador da mensagem.
- Id_operação: identificador da operação, ou seja, indica ao agente receptor qual operação ele deve executar ao receber a mensagem.
- Host: endereço IP da máquina que envia a mensagem.

Figura 39: Formato Geral das Mensagens



O conjunto de mensagem é composto por 9 mensagens, sendo elas destinadas a:

- Criar o aplicativo na máquina distribuída e gerar arquivo de controle.
- Confirmar o recebimento da mensagem para criação de aplicativo.
- Atualizar o endereço IP relacionado com algum aplicativo.
- Informar a criação de um projeto correlato.
- Confirmar a criação do projeto correlato.
- Solicitar ao agente distribuído as atividades para compor uma visão global.
- Confirmar o recebimento da solicitação de envio das atividades, informando a quantidade de atividades e sub-atividades que serão enviadas.
- Responder confirmando ou não o envio das atividades do projeto correlato.
- Envio de atividades.

O detalhamento de cada mensagem e seus campos constam do Apêndice B.

8 APLICAÇÃO DA FERRAMENTA CASE

Este capítulo apresenta a ferramenta CASE implementada a partir da especificação descrita no capítulo 7 e visa comprovar a viabilidade da metodologia proposta neste trabalho apresentado um exemplo de gerenciamento distribuído de projeto, utilizando a ferramenta desenvolvida.

Para a implementação da ferramenta foi utilizada a linguagem de programação Delphi 5.0 em conjunto com o sistema gerenciador de banco de dados Interbase. Para o detalhamento desta ferramenta, denominada GEMETRICS, este capítulo foi dividido de acordo com os grupos de requisitos descritos:

- Requisitos básicos: módulos gerais da ferramenta, implementando cadastros, controles básicos e cálculo da métrica ponto de função.
- Requisitos de apoio à decisão: principais relatórios implementados na ferramenta.
- Requisitos para gerenciamento distribuído: detalha a implementação dos requisitos para viabilizar o gerenciamento distribuído de projetos incluindo o agente proposto.

8.1 Ferramenta CASE GEMETRICS

Visando facilitar a compreensão das partes que compõem a ferramenta desenvolvida, esta seção é descrita na forma de um tutorial passo-a-passo das etapas envolvidas no planejamento.

Passo 1: Inicialmente, para a criação de um novo planejamento relacionado a um aplicativo, este deve ser cadastrado fornecendo dados como: nome, data de criação e se o mesmo é local ou distribuído. Um aplicativo local contém apenas um projeto associado a ele, já o aplicativo distribuído possui mais de um projeto associado. Somente no caso de aplicativo distribuído será fornecida uma visão global (gráfico de Gantt) do planejamento, reunindo os vários projetos relacionados. Vide Anexo 2 tela 1.

Passo 2: Algumas informações a respeito do projeto devem ser informadas, ressaltando-se dados como o modelo de processo de desenvolvimento a ser adotado, gerente responsável e se este projeto cadastrado refere-se ao projeto local ou distribuído (caso o

aplicativo seja distribuído). O conjunto completo de dados trabalhados encontra-se na interface visualizada no Anexo 2 tela 2.

Passo 2.1: Caso o modelo de processo de desenvolvimento desejado e/ou gerente não estejam previamente cadastrados, estes devem ser fornecidos a ferramenta. Referente ao gerente fazem-se necessárias algumas informações básicas como endereço, e-mail, telefone e salário (a tela 3 do Anexo 2 apresenta os dados completos). Quanto ao processo de desenvolvimento devem ser informados o nome fornecido ao modelo e as fases que o compõem (conforme Anexo 2 tela 4).

Passo 3: Uma vez informados todos os dados básicos referente ao projeto, pode-se iniciar o planejamento das atividades a serem desenvolvidas. Para cada atividade prevista deve ser informada a fase do processo de desenvolvimento a que esta atividade está vinculada, bem como a data inicial e a o tempo de desenvolvimento (em dias). Automaticamente a ferramenta executa o algoritmo de PERT/CPM implementado e calcula a data final do projeto, bem como as datas de início e término mais cedo e, datas de início e término mais tarde de cada ferramenta, conforme visualizado na Figura 40. Caso haja uma relação de precedência entre atividades na tela demonstrada na Figura 41 pode-se criar vínculos entre as atividades. Novamente o algoritmo de PERT/CPM é executado e todas as datas são recalculadas. A data final otimizada do projeto também é recalculada a cada alteração nas atividades, esta data é diferenciada da data final caso o planejamento comporte folgas.

Figura 40: Cadastro de Atividades

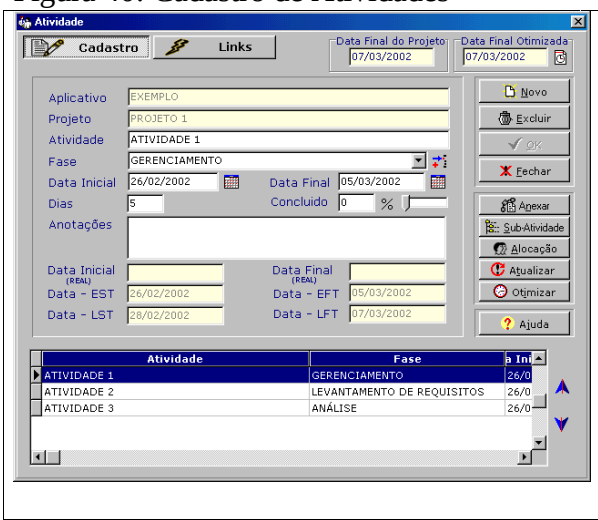
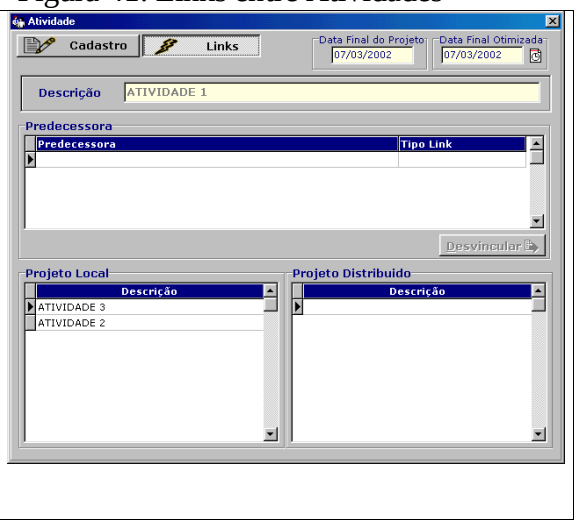


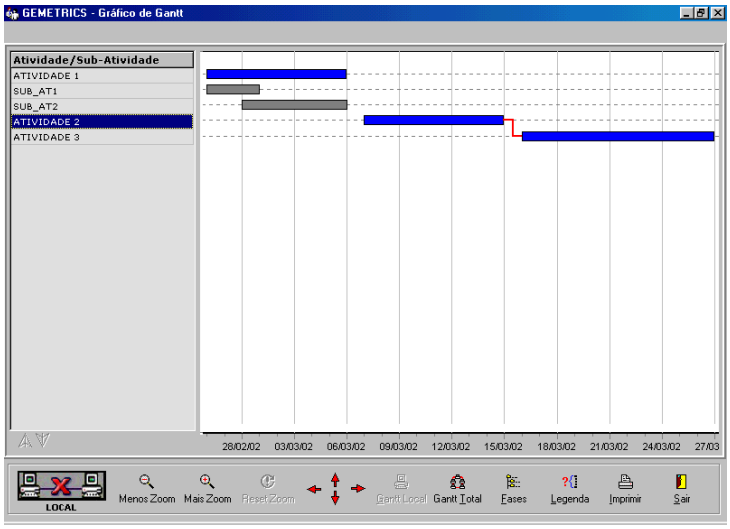
Figura 41: Links entre Atividades



Passo 3.1: No caso de alguma atividade abranger um grande conjunto de tarefas pode-se utilizar a tela 5 visualizada no Anexo 2 para decompor a atividade em sub-atividades.

Passo 4: Para visualizar o planejamento das atividades o gráfico de Gantt foi implementado. A Figura 42 apresenta a forma de visualização do gráfico, onde se tem em azul as atividades planejadas e em cinza as sub-atividades. O gráfico de Gantt também apresenta o vínculo entre atividades, representado através de uma linha entre elas. Vale ressaltar que esta visão apresentada na Figura 42 refere-se a uma visão local.

Figura 42: Gráfico de Gantt



Passo 5: Após o cadastro de atividade deve-se alocar os recursos que estão associados, ou seja, vinculados para a realização da mesma. A tela 6 do Anexo 2 apresenta a interface para realizar este procedimento. No entanto, caso o recurso necessário ainda não esteja cadastrado o mesmo deverá ser informado. Para esta função tem-se a tela 7 do Anexo 2 onde é necessário informar os dados referentes aos valores associados a cada recurso.

Passo 6: O cálculo da métrica ponto de função está dividido nas seguintes etapas:

a) cadastrar as funções do tipo dado (vide seção 4.3.2.3). Para isto faz-se necessário identificar o tipo da função, entre arquivo lógico interno ou arquivo de interface externa. Caso o usuário deseje é possível detalhar os dados elementares relacionando os registros lógicos relacionados e os dados elementares relacionados, ou simplesmente informar a quantidade de dados elementares, sem detalha-los. Após informar os valores associados a ferramenta computa a complexidade da função e a quantidade de pontos de função. A interface desta função se encontra no Anexo 2 tela 8.

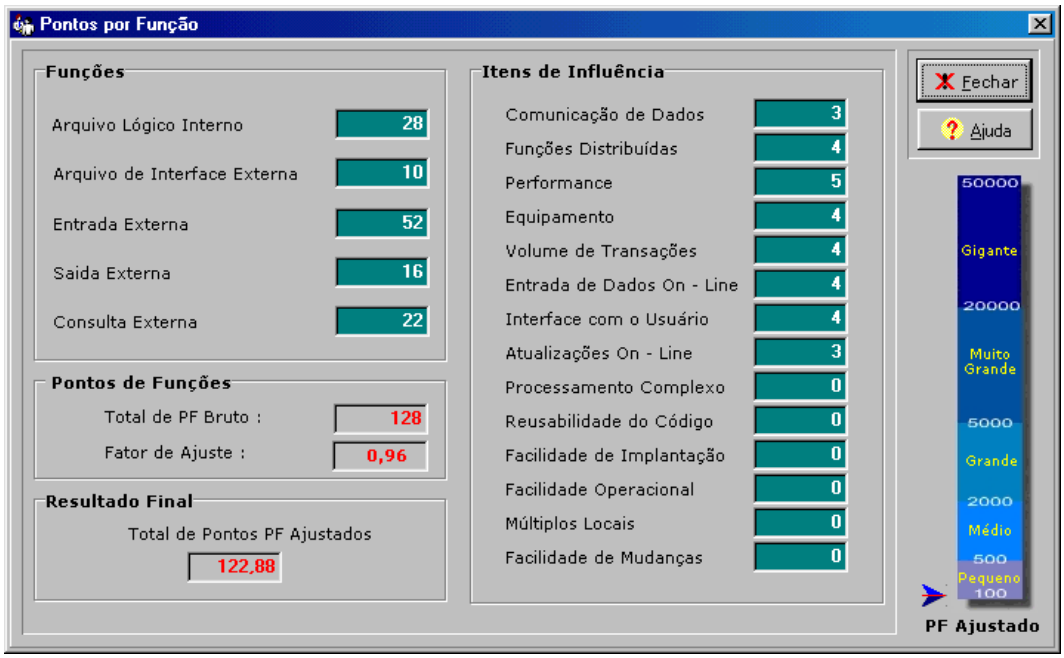
b) cadastrar as funções do tipo transação (conforme apresentado na seção 4.3.2.4). Devendo-se identificar o tipo da função entre: entrada externa, saída externa e

consulta externa. A exemplo da função tipo dado, caso o usuário deseje é possível detalhar os dados elementares ou simplesmente informar a quantidade associada a função. Após informar os valores a ferramenta computa a complexidade da função e a quantidade de pontos de função. A interface desta função es encontra no Anexo 2 tela 9.

c) Após o cadastro das funções se faz necessário informar os fatores de ajustes, conforme descrito na seção 4.3.2.5. A tela 10 do Anexo 2 apresenta a forma implementada para o cálculo do fator de ajuste.

d) Como resultado a tela visualizada na Figura 43 apresenta o total por tipo de função, o valor do fator de ajuste, o valor associado a cada item do fator de ajuste e o total de pontos de função ajustados. Além disso, apresenta uma escala dimensionando o projeto em questão.

Figura 43: Cálculo de Pontos por Função



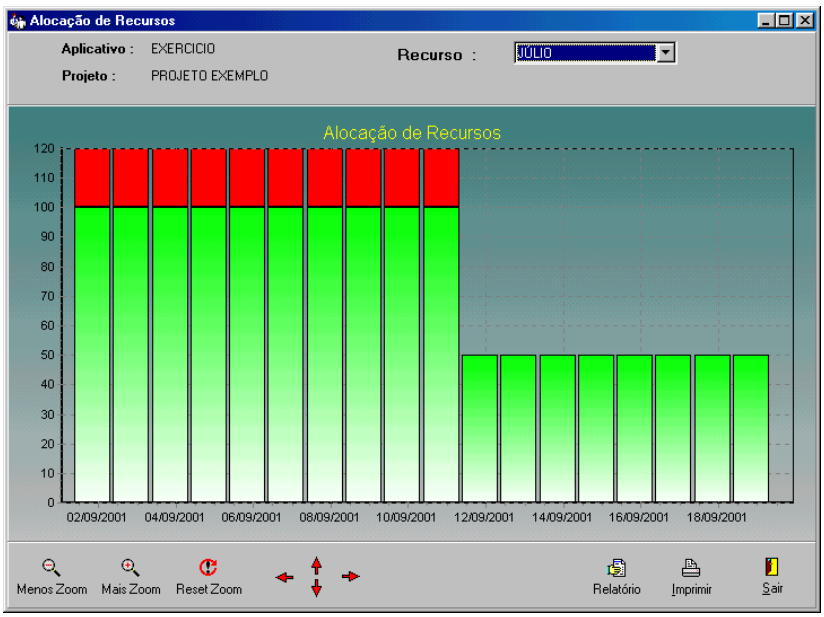
8.2 Relatórios

Esta seção explana a forma de apresentação e os dados que constam de cada relatório previsto na especificação:

- Custo por cargo: apresenta gráfico de colunas representando o valor total em recursos humanos, dividido entre aos cargos alocados para o projeto selecionado, conforme visualizado na tela 11 do Anexo 2.

- Custo por fase: Este relatório apresenta graficamente, através de gráfico de barras, o custo percentual (referente ao valor total do projeto) de cada fase do processo de desenvolvimento.
- Alocação de recursos: apresenta um gráfico relacionando a percentagem de ocupação por dia de cada recurso, conforme visualizado na Figura 44.

Figura 44: Visualizando a Alocação de Recursos



- Perfil profissional por fase: neste relatório, apresentado também sobre a forma de gráfico de linhas, pode-se visualizar o esforço (em horas) empregado por cada cargo em função da fase do processo de desenvolvimento.
- Esforço por fase: este relatório apresenta percentualmente qual o tempo utilizado por cada fase do processo de desenvolvimento, sendo apresentado na forma de um gráfico de barras (vide Anexo 2 tela 14).
- Custo do ponto de função entre projetos: apresenta o valor associado ao desenvolvimento de 1 pontos de função em cada projeto cadastrado, a forma de apresentação encontra-se no Anexo 2 tela 15.
- Tempo de desenvolvimento de um ponto de função entre projetos: igualmente ao item anterior, apresenta o tempo associado ao desenvolvimento de 1 ponto de função para todos os projetos cadastrados, conforme ilustrado no Anexo 2 tela 15.

- Quantidade de pontos de função bruto entre projetos: Apresenta um comparativo entre a quantidade de pontos de função bruto entre as diferentes aplicações planejadas na ferramenta, podendo ser visualizado na tela 16 do Anexo 2.
- Comparativo entre fatores de ajustes dos projetos: apresenta os fatores de ajustes de todos os projetos relacionados na ferramenta em forma de gráfico de linha. O Anexo 2 tela 17 contém a forma de visualização deste relatório.
- Comparativo do tamanho dos sistemas planejados: este relatório apresenta a classificação dos projetos (pequeno, médio, grande, muito grande e gigante) e o tamanho do sistema expresso em pontos de função. Além destes dados é apresentado um gráfico de pizza para expressar o percentual de projetos desenvolvidos em cada classificação, conforme apresentado na tela 18 do Anexo 2.
- Estimativas: apresenta o resumo do planejamento contendo a data de início e de término do projeto, o tempo planejado, e o custo com pessoal e equipamento associado, conforme consta da tela 19 do Anexo 2.

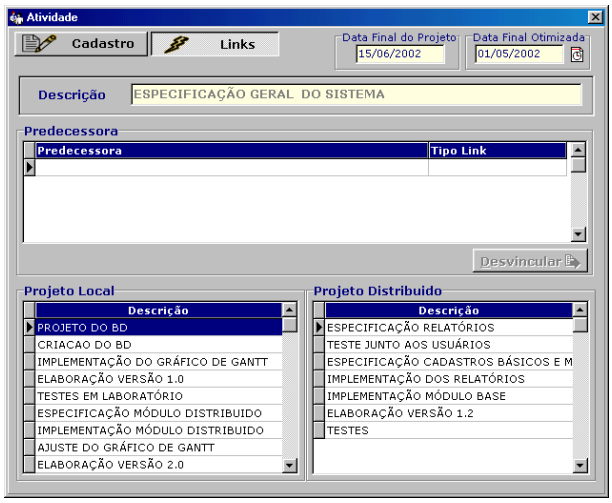
8.3 Gerenciamento distribuído

Esta seção aborda a implementação das funções referentes ao gerenciamento distribuído, detalhando os aspectos incorporados à ferramenta CASE e as funções implementadas pelo agente.

8.3.1 Ferramenta CASE GEMETRICS

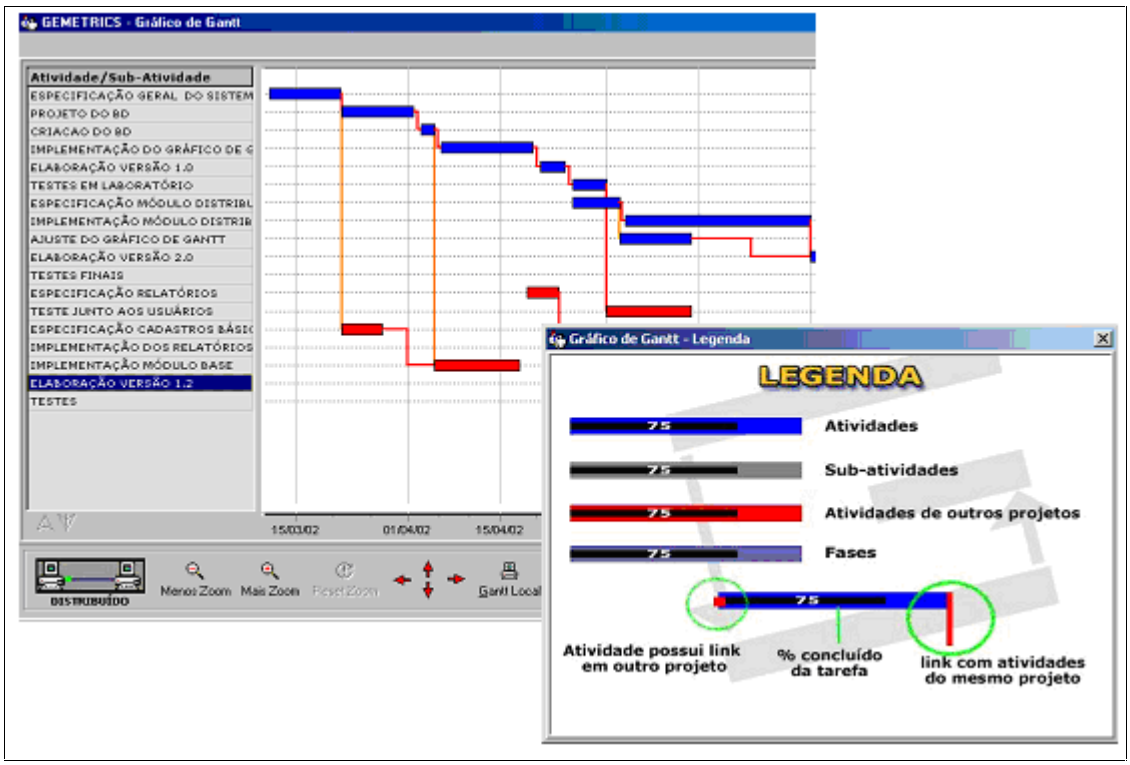
Alguns recursos existentes na ferramenta CASE são específicos para o gerenciamento distribuído. Para iniciar o processo de planejamento distribuído especificando os vínculos entre as atividades dos diferentes projetos deve-se utilizar o botão Atualizar (Figura 40). Com este procedimento o agente recupera todas as atividades dos projetos correlatos e estas ficam disponíveis em caixa específica na tela de *links* entre atividades (conforme Figura 45). Nesta tela o usuário pode criar o vínculo entre atividades do seu projeto ou com qualquer atividade dos projetos correlatos.

Figura 45: Tela para Criação de Links entre Atividades de Diferentes Projetos



Quanto a visualização destas atividades, a metodologia prevê alterações nos gráficos de Gantt ou de PERT para poder visualizar e identificar as diferentes atividades. Na implementação da ferramenta optou-se por adaptar o gráfico de Gantt para permitir o acompanhamento do projeto. Tais alterações foram previstas conforme apresentadas na Figura 46. Ou seja, as atividades dos projetos correlatos são apresentadas diferentemente (no caso, utilizando a cor vermelha).

Figura 46: Adaptação do gráfico de Gantt para o GEMETRICS

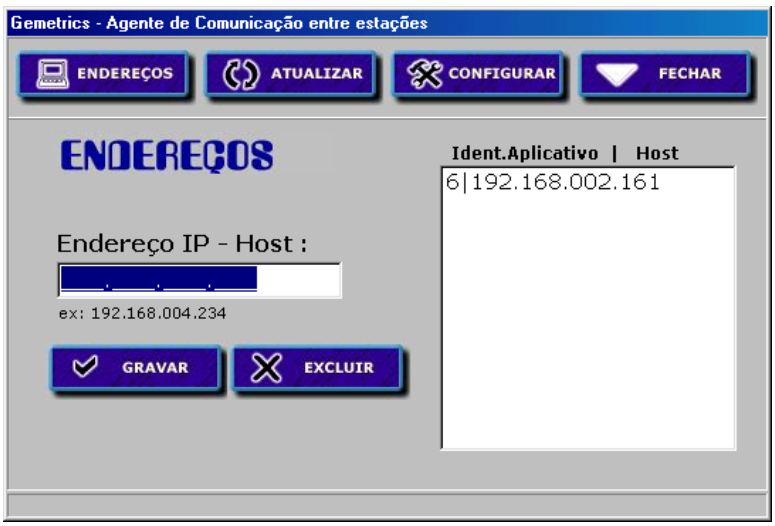


Outra adaptação que se fez necessária deve-se a como indicar na visão local que a atividade possui um link com uma atividade de projeto correlato. Nestes casos, é apresentado ao usuário um ponto vermelho na atividade em questão, conforme ilustra a legenda presente na Figura 46.

8.3.2 Agente

Ao se cadastrar uma aplicação, e esta sendo desenvolvida de forma distribuída, informado na tela 1 do Anexo 2, é apresentado automaticamente ao usuário a interface de configuração do agente (Figura 47). Neste primeiro momento o usuário deve informar o(s) endereço(s) IP referente à máquina onde serão planejados os demais projetos que compõem o aplicativo.

Figura 47: Interface de Configuração do Agente



A interface do agente também possui implementada a opção de atualizar (demonstrada na tela 20 do Anexo 2) que permite acompanhar o envio de mensagens pendentes aos demais agentes. Na opção configurar o usuário pode estabelecer suas preferências quanto ao tempo e forma de atualização das mensagens, podendo também atualizar o seu endereço IP – sendo o mesmo enviado a todos os demais agentes automaticamente (a interface desta opção está retratada na tela 21 do Anexo 2).

Visando a manutenção das informações que são utilizadas freqüentemente pelo agente, 4 arquivos se fazem necessários:

- dis?.gmt – são os arquivos usados para guardar informações imprescindíveis para localização do aplicativo e do agente distribuído, ou seja, este arquivo conterá o identificador do aplicativo local e os identificadores dos aplicativos distribuídos e seus respectivos endereços para localização na rede (endereço IP) informações utilizadas para futuras transferências de informações. O símbolo "?" representa o identificador do aplicativo local, pois cada aplicativo local terá um arquivo dis?.gmt;
- mensag.gmt – este arquivo contém as mensagens enviadas pelo agente local aos demais agentes, essas informações são armazenadas para futuros reenvio de mensagens, se houver necessidade.
- log.gmt – este arquivo contém a identificação de cada mensagem enviada pelo agente local aos agentes distribuídos e a confirmação ou não do recebimento destas mensagens por parte dos agentes distribuídos. Mensagens não enviadas corretamente aos agentes distribuídos, por qualquer que seja o motivo, serão posteriormente enviadas com base nas informações armazenadas neste local.
- reg.gmt – este arquivo contém informações de configuração do agente, tais como:
    - Ativar/Desativar reenvio automático de informações
    - Tempo para reenvio de informações
    - Número de dias que as informações no histórico de mensagens.
    - Como se pode observar na descrição dos arquivos,

8.4 Exemplo de aplicação

Esta seção apresenta um exemplo da aplicabilidade da metodologia prosposta neste trabalho bem como, o uso da ferramenta CASE GEMETRICS para auxiliar na aplicação da metodologia.

Primeiramente, a seção 8.4.1 explora a questão do planejamento de um aplicativo, no caso o desenvolvimento da própria ferramenta CASE GEMETRICS, de forma distribuída. A seção 8.4.2 apresenta a contabilização dos pontos de função de uma parte da ferramenta CASE.

8.4.1 Planejamento da Ferramenta

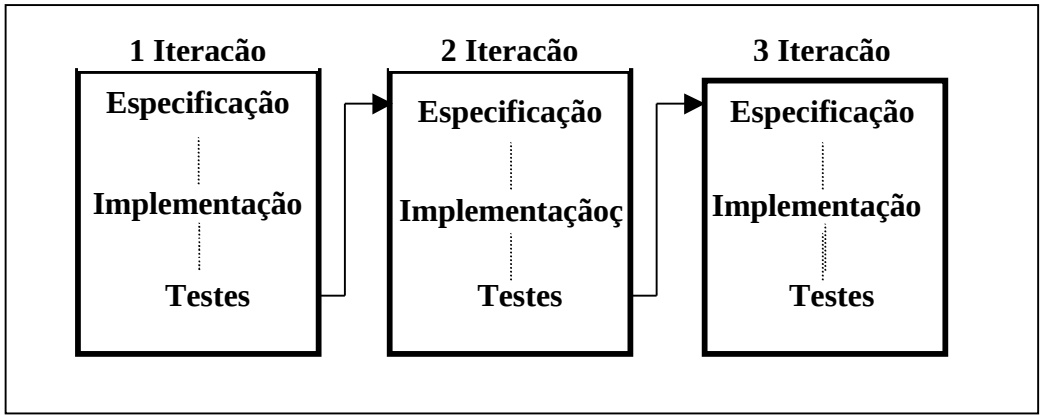
O aplicativo a ser planejado é uma ferramenta CASE para viabilizar o gerenciamento e planejamento distribuído de projetos e integre métrica de software para suporte a decisão. Sendo que o escopo da ferramenta tem como principais funções:

- Cadastros básicos para viabilizar o gerenciamento, tais como recursos, atividades, processo de desenvolvimento, alocação de recursos, gerente responsável, dados dos projetos, etc.
- Implementar a métrica ponto de função.
- Implementar o gráfico de Gantt para acompanhamento do projeto.
- Viabilizar o gerenciamento de aplicativos desenvolvidos de forma distribuída.
- Implementar um conjunto de relatórios fornecendo dados sobre os projetos planejados.

O processo de desenvolvimento adotado para este aplicativo foi baseado no desenvolvimento iterativo, o qual consiste em desenvolver o aplicativo incrementalmente, onde a cada incremento alguma funcionalidade é adicionada até que o sistema completo esteja implementado.

Assim, preveu-se 3 iterações sendo que ao final de cada iteração se teria uma versão funcional da ferramenta. A primeira iteração prevê o desenvolvimento dos cadastros básicos, do gráfico de Gantt e da métrica de software. Com a conclusão destas funções, e duas seções de testes, uma em laboratório e outra junto a usuários, tem-se primeira versão operacional da ferramenta. Numa segunda iteração são acrescentados os relatórios propostos, compondo assim a versão 1.2 da ferramenta. E, por fim, na última iteração (resultando na versão 2.0) é acrescentado o módulo específico para o gerenciamento distribuído, conforme ilustrado pela Figura 48.

Figura 48: Processo de Desenvolvimento da Ferramenta CASE



Para o desenvolvimento deste aplicativo dois projetos foram planejados. Estes projetos possuem gerentes diferentes que devem efetuar o planejamento considerando as atividades envolvidas no seu projeto, no entanto, também devem integrar o planejamento com o projeto correlato que em conjunto dará origem a ferramenta desejada. Assim,os Quadros 22 e 23 representam o planejamento das atividades de cada um dos projeto relacionados.

Quadro 22: Planejamento Projeto 1

Projeto 1						
Id_atividade	**Atividade**	**Fase**	**Duração (dias)**	**Recursos alocados**	**Vínculo (Id_atividade)**	
1	Especificação geral do sistema	Especificação	10	Mário		**1 ITERAÇÃO**
2	Projeto do BD	Especificação	7	Carla	1	
3	Criação do BD	Especificação	2	Carla	2	
4	Implementação do gráfico de Gantt	Implementação	10	Carla	3	
5	Elaboração versão 1.0	Implementação	3	Mário	4, 2 (proj. 2)	
6	Testes em laboratório	Teste	3	Gilberto	5	
7	Especificação módulo distribuído	Especificação	5	Mário		**3 ITERA**
8	Implementação módulo distribuído	Implementação	20	Mário	7	

9	Ajuste do gráfico de Gantt	Implementação	7	Carla	7	**ÇÃO**
10	Elaboração versão 2.0	Implementação	4	Mário	8, 9, 7 (proj. 2)	
11	Testes finais	Teste	5	Mário, Carla	10	

Quadro 23: Planejamento Projeto 2

Projeto 2						
Id_atividade	**Atividade**	**Fase**	**Duração (dias)**	**Recursos alocados**	**Vínculo (Id_atividade)**	**1 ITERAÇÃO**
1	Especificação cadastros básicos e métrica	Especificação	5	Mateus	1 (proj. 1)	
2	Implementação módulo base	Implementação	10	Júnior	1, 3 (proj. 1)	
3	Testes junto aos usuários	Teste	3	Gilberto	6 (proj. 1)	
4	Especificação relatórios	Especificação	3	Mateus		**2 ITERAÇÃO**
5	Implementação dos relatórios	Implementação	5	Mateus, Júnior	4	
6	Elaboração versão 1.2	Implementação	3	Mateus	5	
7	Testes	Teste	3	Mateus, Júnior	6	

Ao se colocar estas atividades, com seus devidos vínculos, na ferramenta CASE GEMETRICS pode-se visualizar graficamente o planejamento efetuado. Assim, a Figura 49 reflete o planejamento do projeto 1 e a Figura 50 apresenta o cronograma planejado no projeto 2.

Figura 49: Visão Local do Projeto 1

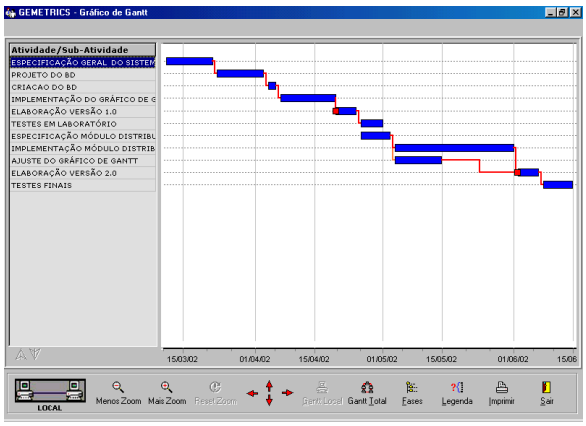
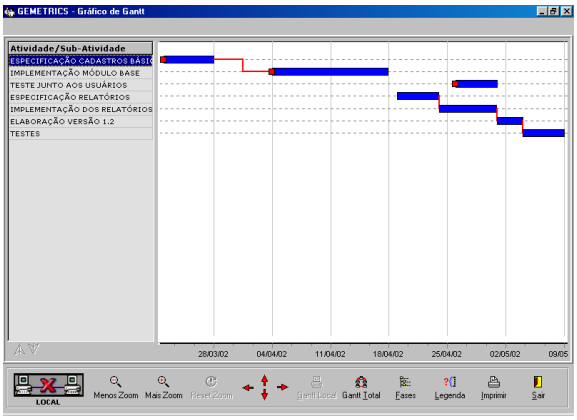
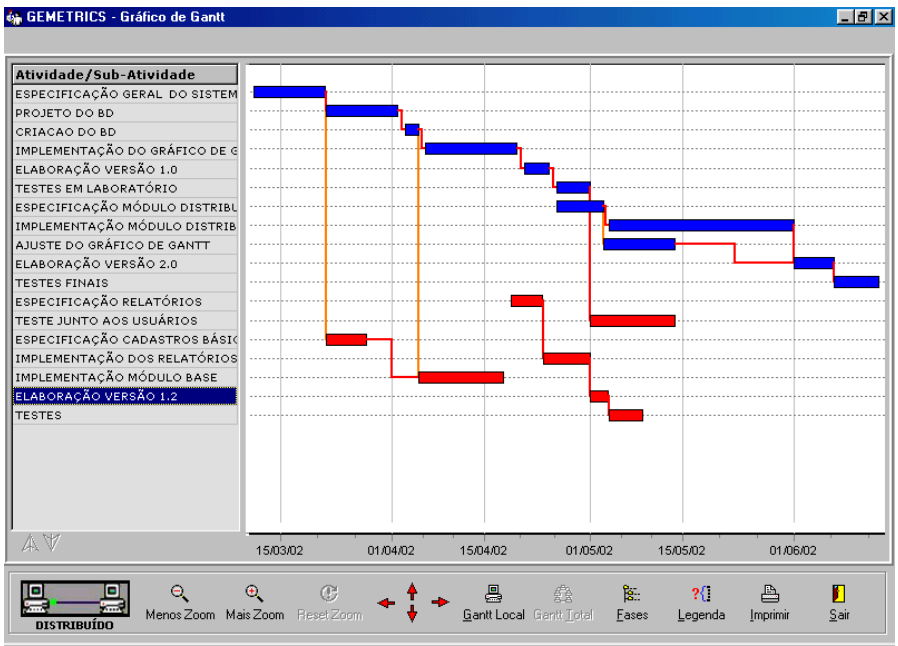


Figura 50: Visão Local do Projeto 2



Pode-se verificar que diversas atividades possuem links com projetos correlatos. Portanto, o planejamento do aplicativo (planejamento do projeto 1 em conjunto com o projeto 2) somente pode ser acompanhada quando vista sobre a visão global, conforme retrata a Figura 51.

Figura 51: Visão Global do Planejamento do Aplicativo



8.4.2 Métrica Pontos por Função

Para ilustrar o emprego da métrica, utilizou-se como base a forma de apresentação exposta em (Braga, 1996), onde primeiramente apresenta uma breve descrição do sistema a ser medido, seguido das principais funções encontradas no

sistema, posteriormente se tem a apresentação dos arquivos e dados considerados na aplicação e se conclui com a contagem dos pontos de função. Braga (1996) também apresenta um esboço da tela a ser implementada para facilitar a contagem, este passo não é apresentado nesta seção pois as telas já foram previamente apresentadas.

Descrição: A ferramenta objetiva apoiar algumas tarefas relacionadas ao gerente de projetos nas suas atividades diárias de planejamento. A ferramenta de gerenciamento deve ser desenvolvida em uma linguagem de quarta geração usando banco de dados e deverá ser usada em um equipamento com sistema operacional Windows. Os usuários deverão despender pouco tempo para aprender a utilizar o software, assim sendo, as telas devem ser de fácil utilização além de terem telas de auxílio associadas.

As principais funções da ferramenta são:

- Cadastramento de gerente
- Alterações no cadastro de gerentes
- Cadastramento de projetos
- Alterações no cadastro de projeto
- Cadastramento de atividades
- Alterações no cadastro de atividades
- Consulta a dados de atividades
- Cadastramento de recursos
- Alterações no cadastro de recursos
- Alocação de recursos
- Gráfico de gantt
- Relatório com o total de horas previstas para cada recurso alocado no projeto.

Assim, os arquivos considerados na contagem são: Gerente, Projeto, Atividade, Recursos e Alocação, contendo os dados descritos abaixo:

GERENTE	
Ger_código	Código do gerente
Ger_descrição	Nome do gerente
Ger_apelido	Apelido do gerente
Ger_enderecoWEB	Endereço do gerente na WEB ou Intranet ou rede
Ger_endereço	Endereço do gerente
Ger_telefone	Telefone do gerente
Ger_email	E-mail do gerente

Ger_nascimento	Data de nascimento do gerente
Ger_valor	Valor do salário mensal do gerente

PROJETO	
Proj_código	Código do projeto
Ger_código	Código do Gerente responsável pelo projeto
Proj_descrição	Nome do projeto
Proj_inicio	Data de início do projeto
Proj_anotações	Anotações sobre o projeto
Proj_autor	Autor do projeto
Proj_assunto	Assunto referente ao projeto
Proj_empresa	Empresa em que ocorre o projeto
Proj_atualizacao	Data da última atualização no projeto

ATIVIDADE	
Ativ_código	Código da atividade
Proj_código	Código do projeto a que pertence a atividade
Ativ_descrição	Nome da atividade
Ativ_datainicial	Data de início da atividade
Ativ_datafinal	Data de término da atividade
Ativ_dias	Quantidade de dias necessários para realização da atividade
Ativ_anotações	Anotações da atividade
Ativ_concluído	Percentual concluído da atividade (0 – 100)
Ativ_dataatualizacao	Data da última atualização
Ativ_datainicial_real	Data que realmente iniciou a atividade
Ativ_datafinal_real	Data que realmente terminou a atividade

RECURSOS	
Recur_codigo	Código do Recurso
Recur_tipo	Tipo do Recurso (PESSOA / EQUIPAMENTO)
Recur_descricao	Nome do Recurso
Recur_taxapadrao	Taxa em R$ para cada hora normal de trabalho (só para PESSOA)
Recur_taxaextra	Taxa em R$ para cada hora extra de trabalho (só para PESSOA)
Recur_valorfixo	Valor fixo do recurso para todo o projeto (só para EQUIPAMENTOS)
Recur_unidades	Unidade disponíveis do recurso em % (100% = 1 unidade)
Recur_observação	Observações sobre o recurso
Recur_horastrabalho	Horas de Trabalho do recurso

ALOCACAO	
Ativ_código	Código da atividade
Recur_código	Código do recurso
Aloc_quantidade	Quantidade alocada do recursos em % (100% = 1 unidade)

Tendo-se listados os dados envolvidos na aplicação pode-se iniciar o processo de contagem.

8.4.3 Arquivos Lógicos Internos

São considerados arquivos lógicos internos um grupo de dados logicamente relacionados ou informações de controle, identificado pelo usuário, mantidos dentro da fronteira da aplicação. Assim, a seguir segue a lista envolvendo a contagem dos arquivos lógicos internos envolvidos na aplicação descrita acima.

- Cadastro de gerente
    - Dados Elementares Referenciados: 9
    - Registros Lógicos Referenciados: 1
    - Grau da Função: simples
- Cadastro de projeto
    - Dados Elementares Referenciados: 8
    - Registros Lógicos Referenciados: 1
    - Grau da Função: simples
- Cadastro de atividade
    - Dados Elementares Referenciados: 10
    - Registros Lógicos Referenciados: 1
    - Grau da Função: simples
- Cadastro de recursos
    - Dados Elementares Referenciados: 9
    - Registros Lógicos Referenciados: 1
    - Grau da Função: simples
- Alocação de recursos
    - Dados Elementares Referenciados: 3
    - Registros Lógicos Referenciados: 1
    - Grau da Função: simples

8.4.4 Arquivos de Interface Externa

Os arquivos de interface externa apresentam a mesma conceituação dos arquivos lógicos internos, no entanto, diferem por serem mantidos dentro da fronteira de outra aplicação**.** Desta forma, a aplicação exemplo não apresenta nenhum arquivo de interface externa, pois não apresenta interfaceamento com outra aplicação.

8.4.5 Entradas externas

Refere-se a um processo elementar que processa dados ou informação de controle que entram pela fronteira da aplicação. Tendo sido identificado na aplicação em questão as seguintes entradas externas:

- Inclusão de gerentes
  - Dados Elementares Referenciados: 10
    - 9 campos e mensagens_erro
  - Arquivos Lógicos Referenciados: 1
  - Grau da Função: simples
- Alteração de gerentes
  - Dados Elementares Referenciados: 10
    - 9 campos e mensagens_erro
  - Arquivos Lógicos Referenciados: 1
  - Grau da Função: simples
- Exclusão de gerentes
  - Dados Elementares Referenciados: 1
    - Somente é necessário selecionar o gerente a ser excluído
  - Arquivos Lógicos Referenciados: 1
  - Grau da Função: simples
- Inclusão de projetos

- Dados Elementares Referenciados: 10
  - 9 campos e mensagens_erro
- Arquivos Lógicos Referenciados: 2
  - Projeto e Gerente (ler informações dos gerentes cadastrados)
- Grau da Função: média

- Alteração de projetos
  - Dados Elementares Referenciados: 10
    - 9 campos e mensagens_erro
  - Arquivos Lógicos Referenciados: 2
    - Projeto e Gerente (ler identificação do gerente)
  - Grau da Função: média
- Exclusão de projetos
  - Dados Elementares Referenciados: 1
    - Somente é necessário selecionar o projeto a ser excluído
  - Arquivos Lógicos Referenciados: 1
  - Grau da Função: simples
- Inclusão de atividades
  - Dados Elementares Referenciados: 12
    - 11 campos e mensagens_erro
  - Arquivos Lógicos Referenciados: 2
    - Atividade e Projeto (ler informação para identificar o projeto no qual a atividade está relacionada)
  - Grau da Função: média
- Alteração de atividades
  - Dados Elementares Referenciados: 12

        - 11 campos e mensagens_erro
    - Arquivos Lógicos Referenciados: 2
        - Atividade e Projeto (identificador do projeto)
    - Grau da Função: média
- Exclusão de atividades
    - Dados Elementares Referenciados: 1
        - Somente é necessário selecionar a atividade a ser excluída
    - Arquivos Lógicos Referenciados: 1
    - Grau da Função: simples
- Inclusão de recursos
    - Dados Elementares Referenciados: 10
        - 9 campos e mensagens_erro
    - Arquivos Lógicos Referenciados: 1
    - Grau da Função: simples
- Alteração de recursos
    - Dados Elementares Referenciados: 10
        - 9 campos e mensagens_erro
    - Arquivos Lógicos Referenciados: 1
    - Grau da Função: simples
- Exclusão de recursos
    - Dados Elementares Referenciados: 1
        - Somente é necessário selecionar a atividade a ser excluída
    - Arquivos Lógicos Referenciados: 1
    - Grau da Função: simples
- Alocação de recursos

- Dados Elementares Referenciados: 4
  - 3 campos e mensagens_erro
- Arquivos Lógicos Referenciados: 3
  - Alocação, Atividade (ler identificador da atividade) e Recurso (ler identificador do recurso)
- Grau da Função: média

8.4.6 Saída Externa

Todo processo elementar que envia dados ou informação de controle para fora da fronteira da aplicação é considerado uma saída externa. Desta forma, as seguintes saídas externas foram consideradas no exemplo em questão:

- Relatório com o total de horas previstas para cada recurso alocado no projeto.
  - Dados Elementares Referenciados: 4
    - Nome do recurso, tipo, taxa padrão, total de horas trabalhadas
  - Arquivos Lógicos Referenciados: 3
    - Recursos (nome do recurso, tipo e taxa padrão), Alocação (quantidade), Atividade (número de dias)
  - Grau da Função: simples

8.4.7 Consulta Externas

Possui a mesma definição da saída externa, diferenciando apenas pelo fato do processamento lógico não conter fórmulas matemáticas ou cálculos, e não criar dados derivados. Assim o única consulta externa é a que se segue:

- Gráfico de Gantt

  Parte de entrada
  - Dados Elementares Referenciados: 1
    - Identificador do projeto

- Arquivos Lógicos Referenciados: 1

Parte de saída

- Dados Elementares Referenciados: 4
  - § Descrição da atividade, data de início, data de término e percentual concluído.
- Arquivos Lógicos Referenciados: 1
- Grau da Função: simples

8.4.8 Cálculo dos Pontos por Função

Uma vez contabilizadas todas as funções do tipo dado e do tipo transação se obter o valor de pontos por função bruto, assim, o Quadro 24 apresenta uma síntese para cálculo do pontos por função bruto da aplicação descrita.

Quadro 24: Cálculo dos Pontos por Função Brutos

Tipo de Função	**Complexidade Funcional**	**Total por Complexidade**	**Total Tipo Função**
Arquivo lógico interno	SIMPLES X 7 = 5 MÉDIA X 10 = COMPLEXA X 15 =	35	35
Arquivo de interface externa	SIMPLES X 5 = MÉDIA X 7 = COMPLEXA X 10 =		
Entrada externa	SIMPLES X 3 = 8 MÉDIA X 4 = 5 COMPLEXA X 6 =	24 20	44
Saída externa	SIMPLES X 4 = 1 MÉDIA X 5 = COMPLEXA X 7 =	4	4
Consulta externa	SIMPLES X 3 = 1 MÉDIA X 4 = COMPLEXA X 6 =	3	3
Total de Pontos de Função Não Ajustados			**86**

Conforme descrito na seção 4.3.2.5 deste trabalho, após o cálculo dos pontos por função brutos deve-se obter o valor do fator de ajuste, tal valor é obtido considerando o grau de influência de 14 fatore. O Quadro 25 apresenta as respostas e o grau de influência de cada fator para o exemplo em questão.

Quadro 25: Fator de Ajuste do Sistema Exemplo

1. Comunicação: aplicação proposta funciona em um computador *stand-alone*.
   Grau de influência: 0
2. Funções distribuídas: aplicação não auxilia na transferência de dados ou funções entre os processadores da empresa.
   Grau de influência: 0
3. Performance: nenhum requerimento especial de performance foi solicitado.
   Grau de influência: 0
4. Equipamento: nenhuma restrição operacional explícita ou mesmo implícita foi incluída.
   Grau de influência: 0
5. Volume de transações: não estão previstos picos de volume de transação.
   Grau de influência: 0
6. Dados on-line: não há entrada de dados on-line
   Grau de influência: 0
7. Interface: possui auxílio a navegação, menu, help e mouse.
   Grau de influência: 2
8. Atualização: atualiza a maioria dos arquivos lógicos internos.
   Grau de influência: 3
9. Processamento complexo: a aplicação não envolve processamento complexo.
   Grau de influência: 0
10. Reusabilidade: não houve preocupação com reutilização de código.
    Grau de influência: 0
11. Facilidade de instalação: não há procedimentos especiais requeridos na implantação do aplicativo.
    Grau de influência: 0
12. Facilidade de operação: o aplicativo minimiza a necessidade de manuseio de papel.
    Grau de influência: 1
13. Múltiplos locais: necessidade de instalar o aplicativo em vários locais foi considerada no projeto, devendo trabalhar em ambientes idênticos de hardware e software.
    Grau de influência: 1
14. Facilidade de mudanças (flexibilidade): neste exemplo não houve nenhuma preocupação específica.
    Grau de influência: 0

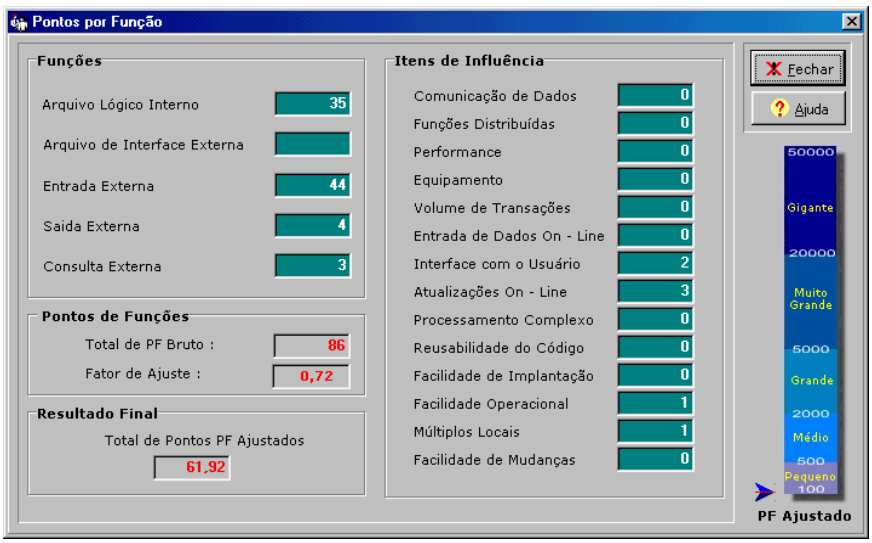
Ao se aplicar a fórmula (descrita na seção 4.3.2.5) para cálculo do fator de ajuste, conforme apresentada abaixo:

Fator de Ajuste = (NIT*0.01)+0.65

tem-se o valor **0.72** como fator de ajuste do sistema em questão. A Figura 52 apresenta a tela final do cálculo da métrica implementada no protótipo da ferramenta CASE desenvolvido, na qual se pode verificar, dentre outros dados, o valor total de

pontos de função ajustados, perfazendo **61.92** pontos de função, o qual segundo a escala determina um sistema pequeno.

Figura 52: Tela do Cálculo Final da Métrica Pontos por Função



8.4.9 Considerações Finais

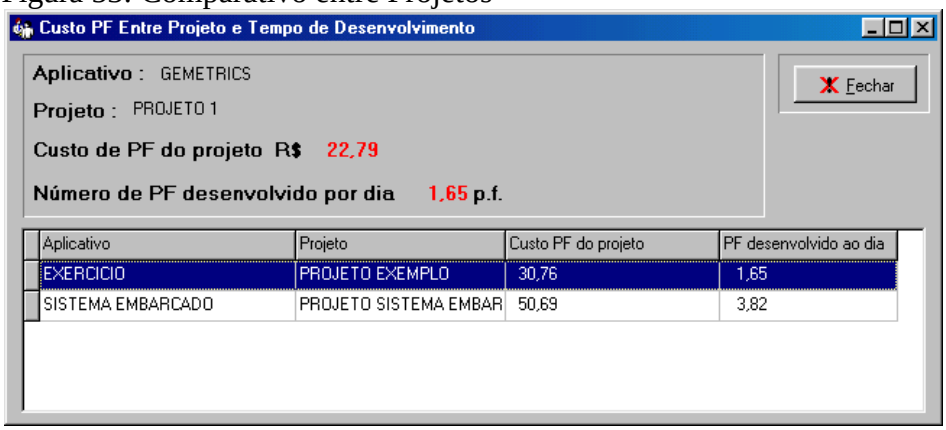
Com o planejamento integrado entre os projetos envolvidos no desenvolvimento de um produto ficam evidenciados os pontos de interação entre os projetos, conforme pode ser visualizado na Figura 51. Assim, tornam-se evidentes as implicações de alterações no planejamento e atrasos no cumprimento de tarefas, viabilizando aos gerentes de projeto acompanhar o andamento no âmbito de todos os projetos correlacionados, tendo, portanto, uma visão mais realista do processo de desenvolvimento como um todo.

A partir do momento que se tem todo o planejamento, com definição das atividades, prazos para desenvolvimento e recursos alocados, e também a contagem dos PF do projeto em questão se pode fazer um comparativo entre os projetos desenvolvidos e efetuar uma avaliação do processo empregado.

Por exemplo, na Figura 53 é ilustrada a tela em que são apresentados os projetos já desenvolvidos com seus valores associados (custo e tempo de desenvolvimento de 1 PF). Pode-se observar que o custo deste projeto está abaixo dos demais projetos já desenvolvidos, o que pode ser interpretado como um amadurecimento do processo de desenvolvimento. Outra constatação possível de ser realizada refere-se a produtividade

da equipe, onde pode-se constatar que tanto o projeto em questão quanto o anteriormente desenvolvido possuem o mesmo valor, ou seja, a produtividade da equipe se manteve constante mesmo com as alterações no processo de desenvolvimento.

Figura 53: Comparativo entre Projetos



E, assim como este exemplo de resultado diversos outros podem ser obtidos considerando os vários relatórios gerados pela ferramenta. No entanto, este tipo de análise somente se torna possível no momento em que se tem um processo maduro de gerenciamento e desenvolvimento de projetos, onde estes resultados obtidos através de métricas de software integrado ao gerenciamento de projetos deverão provocar constantes aperfeiçoamentos no processo de desenvolvimento da organização aumentando a qualidade do processo e dos produtos associados.

9 CONCLUSÕES

No decorrer deste trabalho foram abordados os principais conceitos envolvidos no gerenciamento de projetos e métricas de software. Visando embasar o presente trabalho e estabelecer requisitos para a metodologia proposta, também foi apresentada análise de 4 projetos de pesquisa, 6 ferramentas de gerenciamento de projetos e 4 ferramentas de métrica de software. Também compondo o levantamento de requisitos, foi aplicado um questionário junto às empresas desenvolvedoras de software para tomar conhecimento da realidade das empresas em relação ao gerenciamento de projetos. Tais estudos permitiram constatar o que seria um diferencial do presente trabalho frente aos demais projetos considerados e, também, o que seria necessidade direta dos gerentes de projetos. Como principal resultado obtido destes levantamentos tem-se que o desenvolvimento de uma metodologia para gerenciamento distribuído de projetos se constitui tanto em uma necessidade em termos de mercado quanto um diferencial em termos de produto.

Assim, baseando-se nos estudos apresentados neste trabalho, ficou comprovado que o desenvolvimento desta metodologia e, conseqüentemente, de uma ferramenta de apoio, é um projeto viável tanto ao nível de desenvolvimento/elaboração da metodologia quanto de sua aplicação prática, conforme pode ser constatado nos capítulos 7 e 8 do presente trabalho.

Portanto, se obteve como um dos produtos finais deste projeto uma metodologia que se mostrou adequada a orientar o gerenciamento distribuído de projetos, envolvendo tanto a etapa de planejamento quanto à etapa de monitoramento, considerando aspectos tais como tempo, custo, recursos e atividades envolvidas. Um aspecto importante a ser ressaltado refere-se aos métodos empregados para viabilizar o gerenciamento distribuído, pois tanto a ferramenta de monitoramento (gráfico de Gantt) quanto a técnica de PERT/CPM empregadas puderam ser perfeitamente adaptadas à metodologia proposta. Com isto, pode-se obter maior facilidade na aplicação/uso da metodologia, pois um gerente de projeto experiente tem conhecimento destas técnicas e sabe utilizá-las eficientemente. Portanto, a metodologia proposta não causa uma mudança de paradigma e sim amplia o potencial de utilização de técnicas já amplamente utilizadas e aceitas.

Ainda na busca por um diferencial frente aos projetos e ferramentas pesquisadas, e considerando também os resultados do levantamento de requisitos junto às empresas, foi constatado que o emprego de métrica de software integrado ao gerenciamento de projetos também se constituiria em um diferencial, considerando, principalmente, os benefícios obtidos a partir do seu uso.

Assim, a metodologia indica a viabilidade do emprego de métricas de software, representada na forma de diversos relatórios, a serem empregados tanto na fase de planejamento quanto na fase de monitoramento. E, através da especificação da ferramenta e da implementação apresentada, pode-se verificar diversas formas de se utilizar métrica de software para apoiar ao processo decisório.

Durante os estudos para o desenvolvimento da ferramenta CASE, algumas técnicas foram consideradas para serem empregadas, tais como *groupware* e IA. As características principais de uma ferramenta *groupware* não se enquadravam nos requisitos a serem implementados pela ferramenta CASE. No entanto, considerando a área de Inteligência Artificial Distribuída, a qual apresenta o conceito de agentes, se enquadrou perfeitamente para com os requisitos necessários, principalmente ao se estudar a arquitetura baseada em troca de mensagens. Portanto, a ferramenta CASE foi desenvolvida com vista às características de agentes para viabilizar o gerenciamento distribuído de projetos.

Por fim, os benefícios da utilização da metodologia apresentada neste trabalho poderão ser obtidos tanto pelas empresas tradicionais, as quais desenvolvem projetos de forma distribuída, quanto pelas organizações virtuais. A definição de Travica (1997) para uma organização virtual evidencia o potencial de uso da metodologia por este tipo de empresa:

> "uma organização virtual se refere à coleção temporária ou permanente de indivíduos, grupos, ou unidades organizacionais dispersas geograficamente – que pertençam ou não a uma mesma organização – ou organizações no seu todo que dependem de *links* eletrônicos com a finalidade de completar seu processo de produção".

Os *links* eletrônicos citados por Travica (1997) podem ser guiados pela metodologia proposta e suportado através de uma ferramenta como a especificada neste trabalho.

Concluindo, este trabalho apresentou e embasou a necessidade de se ter uma metodologia, suportada por uma ferramenta, que auxilie os gerentes de projetos de software no planejamento e gerenciamento de projetos desenvolvidos de forma distribuída. Esta metodologia e a ferramenta foram contempladas no decorrer do trabalho e apresentadas em detalhes para viabilizar a sua utilização, incluindo também um exemplo de sua aplicação. Por fim, tem-se evidenciado, nesta conclusão, os benefícios e os campos de atuação aos quais este trabalho se enquadra. No entanto, diversos outros trabalhos surgem a partir da realização deste, dentre os quais constam os relacionados na seção 9.1.

9.1 Trabalhos Futuros

Este trabalho tem como objetivo principal o desenvolvimento de uma metodologia para conduzir o gerenciamento distribuído de projetos. Neste sentido, a tecnologia utilizada para viabilizar o desenvolvimento de uma ferramenta CASE foi à tecnologia de agentes. No entanto, certamente pode-se envolver também o uso de Raciocínio Baseado em Casos (RBC) para auxiliar no processo decisório, inclusive utilizando os resultados obtidos da métrica de software prevista na metodologia. Alguns exemplos do emprego de RBC ao gerenciamento de projetos podem ser vislumbrados nos projetos de pesquisa, apresentados na seção 3.4 deste trabalho, o que atesta a viabilidade deste como um trabalho futuro.

A metodologia proposta prevê a utilização tanto do gráfico de Gantt quanto do PERT para visualizar o planejamento e monitorar o andamento de um projeto. Neste trabalho foi empregado o gráfico de Gantt para comprovar a viabilidade deste recurso. No entanto, pode-se como trabalho futuro adaptar e implementar o gráfico de PERT para apresentar uma visão global do planejamento em caso de projeto distribuído.

Este trabalho visa apenas o gerenciamento distribuído de projetos em nível de gerencia. Ou seja, um planejamento integrado das atividades relativas aos projetos. Pode-se também desenvolver uma metodologia, e conseqüentemente recursos

desejáveis em uma ferramenta CASE, visando facilitar o gerenciamento da equipe de desenvolvimento. Prevendo, por exemplo, o gerenciamento de documentos, a distribuição de atividades, o acompanhamento do andamento das atividades, dentre outros recursos.

Outra área para expandir o presente trabalho trata-se de expandir a metodologia visando enquadramento em normas de gerenciamento de projetos, tais como as previstas pelo CMM, ISO ou PMI. Por exemplo, o PMI possui um documento denominado PMBOK que sub-divide o gerenciamento de projetos em 9 áreas de conhecimento. Atualmente, a metodologia proposta atua principalmente no gerenciamento distribuído tendo como foco o tempo de desenvolvimento, enfim, poderia-se fazer um estudo para atender as outras áreas do conhecimento previstas no PMBOK.

Como se pode notar, diversas são as áreas possíveis para dar continuidade ao presente trabalho, pois esta é uma área de extrema importância para o sucesso dos projetos desenvolvidos e ferramentas que suportem suas atividades ainda são poucas quando comparadas com as destinadas a automatizar as fases do processo de desenvolvimento.

Assim, esta seção deixa sugestões para continuidade deste trabalho contemplando áreas de pesquisa vinculadas a IA, engenharia de software e qualidade de software.

10 REFERÊNCIAS BIBLIOGRÁFICAS

AGUIAR, M. **Pontos de Função Aplicados a Tecnologias Novas e Emergentes.** Disponível em: <http://www.bfpug.com.br/>. Acesso em: 10 ago. 2000.

AGUIAR, Maurício. Developer's Connection. **Developers' Magazine**, Rio de Janeiro, ano 5, n. 55, p. 42-43, mar. 2001.

AUER, K. **Agents.** 1995. Disponível em: <http://www.pcug.org.au/~kauer/project/main.htm>. Acesso em: 12 mar. 2001.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: Guia do Usuário.** Rio de Janeiro: Campus, 2000. ISBN 85-352-0562-4.

BOEHM, B. **Software Engineering Economics**. Prentice-Hall, 1981. ISBN 0-13-822122-7

BOMFIM, F.; AZEVEDO M.; HUDSON S. Métricas de Software. Disponível em : <http://www.internext.com.br/mssa/medidas.html>. Acesso em: 06 abr. 2000.

BRAGA, A. **Análise de Pontos de Função**. IBPI Press, 1996. ISBN 85-733-1002-2

CANDÉAS, A.J.; LOPES, C.C.N. Estimativa: Uma ferramenta para agilizar o dimensionamento de projetos no SERPRO com base na metodologia de análise de pontos por função. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE. Caderno de Ferramentas, 13, 1999, Florianópolis. **Anais...** Florianópolis: UFSC, 1999. p 9-12.

CARVALHO, A.M.B.R.; CHIOSSI, T.C.S. Introdução à Engenharia de Software. Campinas: Unicamp Editora, 2001.

CHANG, C. K.; CHRISTENSEN, M. A Net Practice for Software Project Management. **IEEE SOFTWARE**. V. 16, n. 6, p. 80-88, nov./dec. 1999.

COSTA, M.T.C. **Uma Arquitetura Baseada em Agentes para Suporte ao Ensino à Distância**. Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. Florianópolis, 1999.

FENTON, N. E.; PFLEEGER, S. L. **Software Metrics: A Rigorous & Practical Approach.** 2 ed. Boston: PWS Publishing, 1997. ISBN 0-534-95425-1.

FRANKLIN, S.; GRAESSER, A. **Is it an Agent, or just a Program?:** A Taxonomy for Autonomous Agents, University of Memphis, 1996. Disponível em: <:http://www.msci.memphis.edu/~franklin/AgentProg.html>. Acesso em: 02 abr. 2001.

GHEZZI, C.; JAZAYER, M.; MANDRIOLI, D. **Fundamentals of software engineering.** New Jersey: Prentice-Hall, 1991. ISBN 0-13-820432-2.

GIESE, L.F. **Estrutura de Agentes para os Processos de Compra e Venda Utilizando Tomada de Decisão Difusa**. Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Ciência da Computação da Universidade Federal de Santa Catarina, Florianópolis, SC, 1998.

HAZAN, C. **Análise de Pontos por Função: Uma Abordagem gerencial**. CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. Jornadas de Atualização em informática, Mini-curso, 20. Curitiba, 2000.

IFPUG - International Function Point Users Group. Disponível em: <http://www.ifpug.org/>. Acesso em: 06 abr. 2000.

JALOTE, P. **An integrated approach to software engineering.** 2 ed. New York: Springer-Verlag, 1997. ISBN 0-387-94899-6.

JENNINGS, N. R. Agent Software**. Proceedings UNICOM Seminar on Agent *Software***, Londres, UK, 1995

KNOTTS, G.; DROR M.; HARTMAN, B. A **Project Management Tool for Computer-Supported Cooperative Work During Project Planning.** Thirty-First Annual Hawaii International Conference on System Sciences. V. 1: Collaboration Systems and Technology. Koala Coast, 6-9 jan. 1998.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA. **Qualidade e Produtividade no setor de Software Brasileiro.** Brasília, 2000. n. 3, p. 63.

MARTIN, J.; ODELL, J.J. **Análise e Projeto Orientados a Objeto.** São Paulo: Makron Books, 1995. ISBN 85-346-0426-6

O'CONNOR, R.; JENKINS, J. **Using Agents for Distributed Software Project Management.** IEEE 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. Palo Alto, 16-18 jun., 1999.

O'CONNOR, R. **An Architecture for an Intelligent Assistant System for use in Software project Planning.** Ph.D. Thesis, City University. Londres, 2000.

PRESSMAN, R. S. **Engenharia de software.** São Paulo: Makron Books, 1995. ISBN 85-346-0237-9.

Reed, Paul R. Jr. **Desenvolvendo Aplicativos com Java e UML.** São Paulo: Makron Books, 2001.

Ribu, Kirsten. **Estimating Object-Oriented Software Projects with Use Cases.** Tese de Doutorado apresentada no Departamento de Informática da Universidade de Oslo. Oslo, 2001.

ROSENBERG, D. **Use Case Driven Object Modeling with UML: a practical approach.** Massachusetts: Addison Wesley Longman, 1999. ISBN 0-201-43289-7.

ROYCE, W. **Software Project Management: a unified framework.** USA: Addison-Wesley Publishers, 1998. ISBN 0-201-30958-0.

SOMMERVILLE, I. **Software Engineering**. 4 ed. USA: Addison-Wesley Publishers, 1992. ISBN 0-201-56529-3.

SOMMERVILLE, I. **Software Engineering**. 5 ed. USA: Addison-Wesley Publishers. 1996.

SOUZA, E.M.S. **Uma estrutura de agentes para assessoria na Internet.** Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. Florianópolis, 1996.

SOFTWARE PRODUCTIVITY RESEARCH. Disponível em: <http://www.spr.com/>. Acesso em: 05 abr. 2000.

STRAUSS, R. **Managing Multimedia Projects.** USA: Butterworth-Heinemann, 1997. ISBN 0-240-80244-6.

SURJAPUTRA, R.; MAHESHWARI, P. **A Distributed Software Project Management Tool.** IEEE 8th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. Palo Alto, 16-18 jun., 1999.

TRAVICA, B. The Design of the Virtual Organization: A Research Model. In: Association for Information Systems. Americas Conference Indianopolis. **Proceedings...** Indiana, 15-17 Ago, 1997.

VON MAYRHAUSER, A. **Software Engineering: methods and management.** San Diego: Academic Press, 1990. ISBN 0-12-727320-4.

WOOLDRIDGE, M., JENNINGS, N. **Intelligent Agents: Theory and Practice.** Knowledge Engineering Review, Vol. 11, No 2, 1995.

WU, C.; SIMMONS, D.B. Software Project Planning Associate (SPPA): A Knowledge-Based Approach for Dynamic Software Project Planning and Tracking In: IEEE Proceedings of the Twenty-Fourth Annual International Computer Software and Applications Conference, 2000.

APÊNDICES

Esta seção apresenta os apêndices que se fazem necessários para maior compreensão da ferramenta CASE desenvolvida, compondo-se dos apêndices:

Apêndice A: Especificação dos casos de uso que compõem os digramas de caso de uso apresentados na seção 7.2.2.

Apêndice B: Especificação das mensagens utilizadas pelos agentes, conforme descrito na seção 7.2.3.

Apêndice A

Especificação dos Casos de Uso

Nome do Caso de Uso: **Cadastra Atividades**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : usuário cadastra as atividades que compõem as fases de um determinado projeto.

Fluxo principal
1 - usuário define o nome da atividade.
2 – usuário define data de início da atividade.
3 - usuário define duração da atividade (em dias).
4 - sistema calcula a data de término.
5 - usuário define atividade predecessora.
6 - usuário define o dependência com a atividade predecessora:
7 - usuário define restrições
8 - usuário define anotações sobre a atividade
9 - usuário grava informações no banco de dados

Fluxo Alternativo 1:
1 - Não é obrigatória a ocorrência do evento 3, ao invés usuário informa data de término.
2 – Sistema calcula a duração da atividade (em dias).

Fluxo Alternativo 2:
1 - usuário define porcentagem de conclusão da atividade.

Nome do Caso de Uso: **Cadastra Sub-Atividades**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : usuário cadastra as sub-atividades que compõem as atividades de um determinado projeto.

Fluxo principal
1 - usuário define o nome da sub-atividade
2 – usuário define data de início da sub-atividade
3 - usuário define duração (em dias) da sub-atividade
4 - sistema calcula a data de término.
5 – sistema verifica compatibilidade com informações da atividade
6 - usuário define anotações sobre a sub-atividade
7 - usuário define porcentagem de conclusão da sub-atividade
8 - usuário grava informações no banco de dados

Fluxo Alternativo 1:
1 - Não é obrigatória a ocorrência do evento 3, ao invés usuário informa data de término.
2 – Sistema calcula a duração da atividade (em dias).

Fluxo Alternativo 2:
1 - usuário define porcentagem de conclusão da sub-atividade.

Nome do Caso de Uso: **Cadastra Recursos**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Cadastrar todos os recursos que serão utilizados em um projeto.

Fluxo Principal
1 - usuário define o nome do recurso
2 – usuário define o tipo de recurso (pessoa ou equipamento)
3 - usuário define o custo do recurso
    - taxa padrão

- taxa de horas extras
5 - usuário define o cargo do recurso
4 - usuário define unidades disponíveis
6 - usuário define observações sobre o recurso
7 - usuário define horas trabalhadas por dia do recurso
8 - usuário grava informações no banco de dados

Fluxo Alternativo:
1 – Se ação 2 tiver como resposta “equipamento”, definir valor fixo do recurso para o projeto.
2 – A ação 3 e 4 só é executada quando a resposta a ação 7 for “pessoa”.

Nome do Caso de Uso: **Cria Projeto**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : O usuário cria um novo arquivo para gerenciamento de um projeto de software.

Fluxo Principal
1 - usuário define nome para projeto
2 - usuário seleciona modelo a ser utilizado no projeto.
3 - usuário define data de início de projeto
4 - usuário define anotações sobre o projeto.

Fluxo Alternativo
1 - 0o usuário poderá optar por não utilizar um modelo pré-estabelecido (ação 2), cadastrando um novo projeto <<extend>>.

Nome do Caso de Uso: **Aloca Recursos**
Categoria: Realiza Gerenciamento de Projeto
Objetivo: Definir quais recursos serão utilizados por determinada tarefa (atividade) e qual a intensidade de uso destes recursos.

Fluxo Principal
1 - usuário seleciona a tarefa e o recurso que serão alocados.
2 - usuário define a quantidade que este recurso estará disponível para esta determinada tarefa.
3 - usuário grava esta informação no banco de dados.

Nome do Caso de Uso: **Define Parâmetros do Projeto**
Categoria: Realiza Gerenciamento de Projeto
Objetivo: Definir características básicas do projeto e do ambiente de trabalho (ferramenta).

Fluxo Principal
1 - usuário define (altera) título projeto
2 - usuário define assunto do projeto
3 - usuário define autor do projeto
4 - usuário define empresa responsável pelo projeto
5 - usuário define se o projeto é local ou distribuído
6 - usuário grava informações no Banco de Dados

Nome do Caso de Uso: **Cadastra Gerentes**
Categoria: Realiza Gerenciamento de Projeto

Objetivo : Cadastrar todos os responsáveis pelo gerenciamento dos projetos em questão.

Fluxo Principal
1 - usuário define o nome do gerente
2 - usuário define iniciais de identificação do gerente
3 - usuário define endereço http do gerente
4 - usuário cadastra endereço do gerente
5 - usuário cadastra telefone do gerente
6 - usuário cadastra e-mail do gerente
7 - usuário define data de nascimento do gerente
8 – usuário define salário mensal do gerente
9 - usuário confirma inclusão das informações no banco de dados

---

Nome do Caso de Uso: **Cadastra Cargos**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Cadastrar todos os cargos que serão utilizados em um projeto.

Fluxo Principal
1 - usuário define o nome do cargo
2 - confirma inclusão da informação no banco de dados

Nome do Caso de Uso: **Cadastra Fases**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Cadastrar as fases que compõem os modelos previamente cadastrados.

Fluxo Principal
1 - usuário entra com a descrição para cada fase do modelo
2 - usuário define ordem das fases
3 - usuário define anotações sobre a fase

Nome do Caso de Uso: **Cadastra Modelos**
Categoria: Realiza Gerenciamento de Projeto
Objetivo - Cadastrar os tipos de modelos para implementação de software que serão utilizados.

Fluxo Principal:
1 - usuário entra com o nome para o modelo
2 - usuário define anotações dos modelos
3 - usuário cadastra Fases <<include>>
4 - usuário confirma inclusão de modelo no banco de dados.

Nome do Caso de Uso: **Cadastra Aplicativos**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Cadastrar os aplicativos que englobam os projetos distribuídos ou locais.

Fluxo Principal
1 - usuário define o nome do aplicativo
2 - usuário define se aplicação é distribuída ou local
3 - usuário define data de registro de aplicação
4 - o sistema grava as informações no banco de dados

Nome do Caso de Uso: **Abre Aplicativo**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Abrir um aplicativo específico para efetuar alterações.

Fluxo Principal
1 - usuário define o nome do aplicativo a ser aberto
2 – sistema consulta na base de dados informações relativas ao aplicativo informado.

Nome do Caso de Uso: **Visualiza Gráfico de Gantt (Local)**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Mostrar o andamento do projeto com as fases, atividades, sub-atividades, com uma visão limitada as tarefas locais.

Fluxo Principal:
1 - usuário solicita visualização do Gráfico de Gantt Local
2 - o sistema busca no banco de dados as seguintes informações:
- fases
- atividades
- data inicial
- data final
- total completado
- recursos alocados
- tarefa predecessora

3 - sistema retorna o Gráfico de Gantt

Fluxo Alternativo:
1 - se necessário usuário imprime gráfico de Gantt

Nome do Caso de Uso: **Visualiza Gráfico de Gantt (Global)**
Categoria: Realiza Gerenciamento de Projeto
Objetivo : Mostrar o andamento do projeto com os seus processos, fases, atividades, com uma visão geral de todos os gerentes envolvidos.

Fluxo Principal:
1 - usuário solicita visualização do Gráfico de Gantt global
2 - o sistema mostra a última atualização.
3 - o sistema busca no banco de dados (o projeto local e entre os correlatos) as seguintes informações:
- fases
- atividades
- data inicial
- data final
- total completado
- recursos alocados
- tarefa predecessora

4 - sistema retorna o Gráfico de Gantt

Fluxo Alternativo:
1 - se necessário usuário imprime gráfico de Gantt

Nome do Caso de Uso: **Define Complexidade**
Categoria: Calcula Pontos de Função
Objetivo: Tendo identificado o número de ALR e DER para cada uma das funções dos cinco grupos de funções, deve-se determinar a complexidade de cada função.

Fluxo principal:
1 – Ler os RLR e DER informados pelo usuário ou contar a partir do cadastro de dados elementares.
2 - com base nas tabelas a seguir identificar a complexidade de cada função dos 5 grupos.

Arquivos de Interface Externa – AIE

	1 a 19 DER	20 a 50 DER	51 ou mais DER
1 RLR	Simples	Simples	Média
2 a 5 RLR	Simples	Média	Complexa
6 ou mais RLR	Média	Complexa	Complexa

Arquivos Lógicos Internos – ALI

	1 a 19 DER	20 a 50 DER	51 ou mais DER
1 RLR	Simples	Simples	Média
2 a 5 RLR	Simples	Média	Complexa
6 ou mais RLR	Média	Complexa	Complexa

Entradas Externas

	1 a 4 DER	5 a 15 DER	16 ou mais DER
0 ou 1 ALR	Simples	Simples	Média
2 ALR	Simples	Média	Complexa
3 ou mais ALR	Média	Complexa	Complexa

Saídas Externas

	1 a 5 DER	6 a 19 DER	20 ou mais DER
0 ou 1 ALR	Simples	Simples	Média
2 a 3 ALR	Simples	Média	Complexa
4 ou mais ALR	Média	Complexa	Complexa

Consultas Externas

	1 a 4 DER	5 a 15 DER	16 ou mais DER
0 ou 1 ALR	Simples	Simples	Média
2 ALR	Simples	Média	Complexa
3 ou mais RLR	Média	Complexa	Complexa

Nome do Caso de Uso: **Calcula PF bruto**
Categoria: Calcula Pontos de Função
Objetivo: Através dos resultados obtidos no use case "Define Complexidade" , que determinam a complexidade de cada item dos 5 grupos, é calculado o total de pontos de função bruto da aplicação
Pré-Requisito: o usuário deve ter definido as funções do tipo dado e transação.

Fluxo Principal

1 – Ler as funções cadastradas com sua complexidade e cruzar com as tabela abaixo:

Arquivos Lógicos Internos – ALI

Grau de Complexidade	Pontos de função brutos
Simples	7
Média	10
Complexa	15

Arquivos Interface Externa - AIE

Grau de Complexidade	Pontos de função brutos
Simples	5
Média	7
Complexa	10

Entradas Externas - EE

Grau de Complexidade	Pontos de função brutos
Simples	3
Média	4
Complexa	6

Saídas Externas - SE

Grau de Complexidade	Pontos de função brutos
Simples	4
Média	5
Complexa	7

Consultas Externas - CE

Grau de Complexidade	Pontos de função brutos
Simples	3
Média	4
Complexa	6

2 - Somar o total de PF dos 5 tipo de função a fim de chegar ao total de pontos não ajustados.
3 - Gravar no banco de dados o PF bruto de cada um dos 5 tipos de funções

---

Nome do Caso de Uso: **Definir Fatores de Ajustes**
Categoria: Calcula Pontos de Função
Objetivo: calcular o fator de ajuste baseado em 14 características gerais do sistema. Cada característica está associada a descrição que auxilia na determinação do nível de influência de cada uma. Os níveis de influência variam de zero a cinco, respectivamente representam nenhuma influência até influência forte.

Fluxo Principal
1 - o usuário define o nível de influência do software/projeto em questão para cada uma das 14 características gerais do sistema.
2 - calcular o nível de influência (NI) através da soma dos pesos de cada uma das 14 características.
3 - calcular o fator de ajuste através da equação:

Fator-ajuste=(NI * 0,01) + 0,65

Nome do Caso de Uso: **Calcula PF ajustado**
Categoria: Calcula Pontos de Função
Objetivo: Calcular o número de PF ajustados, através dos PF bruto e o Fator de Ajuste.
Pré-Requisito: O usuário ter cadastro as funções (para cálculo do PF bruto) e ter definido os fatores de ajuste.

Fluxo Principal:
1 - Definir o ponto de função ajustado através da seguinte função:
PF ajustado = PF bruto * Fator de ajuste
2 - Gravar no Banco de Dados o total de PF ajustado.

Nome do Caso de Uso: **Informa Tempo/PF**
Categoria: Calcula Pontos de Função
Objetivo: este indicador mostra o tempo para se desenvolver um ponto de função na aplicação em questão. Informações obtidas através da soma total do tempo de desenvolvimento do projeto dividido pelo número total de PF deste projeto.

Fluxo Principal
1 - o sistema busca as informações do tempo total de desenvolvimento do projeto e Total de PF ajustados
2 - o sistema guarda no banco de dados do respectivo projeto em avaliação, o valor de tempo necessário para desenvolvimento de 1 PF:
Resultado = Tempo Total de desenvolvimento / Soma Total de PF ajustado
3 – o usuário visualiza a informação

Nome do Caso de Uso: **Informa $/PF**
Categoria: Calcula Pontos de Função
Objetivo: este indicador mostra o preço para se desenvolver um ponto de função na aplicação em questão. Informação obtida através da soma de todos os gastos envolvidos no desenvolvimento da aplicação dividido pelo número total de PF desta aplicação.

Fluxo Principal
1 - o sistema busca a informação do valor total de desenvolvimento do projeto e Total de PF ajustados
2 – o sistema calcula o valor através da seguinte fórmula:
Resultado = Valor Total de Desenvolvimento (R$) / Soma Total de PF ajustados
3 – o usuário visualiza a informação

Nome do Caso de Uso: **Verifica Tamanho do Projeto**
Categoria: Calcula Pontos de Função
Objetivo: identificar o tamanho dos projetos produzidos na instalação.

Fluxo Principal
1 - o sistema busca as informações do Total de PF ajustados do projeto.
2 - o sistema classifica o tamanho do projeto de acordo com a tabela a seguir:

Entre (PF)	
100 a 500	Pequeno
500 a 2000	Médio
2000 a 5000	Grande
5000 a 20000	Muito Grande
20000 a 50000	Gigante

3 – o usuário visualiza a informação

Nome do Caso de Uso: **Define Funções do Tipo Dado**
Categoria: Calcula Pontos de Função
Objetivo: Manter cadastrada as funções do tipo dado pertinentes ao sistema em questão.

Fluxo Principal:
1 – usuário cadastra descrição da função
2 – usuário cadastra tipo de função de dado (ALI ou AIE)
3 - usuário cadastra dados elementares <<extend>> para cada função.
4 - sistema calcula dados elementares referenciados e registros lógicos referenciados para cada função

Fluxo Alternativo:
1 – se tipo de função (ação 2) é igual EE ou SE, definir se função é pertinente ou não ao aplicativo em questão.
   Se ação 1 não realizada
2 - usuário define funções referentes as Entradas Externas
      - dados elementares referenciados
      - registros lógicos referenciados ou arquivos lógicos referenciados.
3 - usuário define funções referentes as Saídas Externas
      - dados elementares referenciados
      - registros lógicos referenciados ou arquivos lógicos referenciados.
4 - usuário define funções referentes as Consultas Externas
      - dados elementares referenciados
      - registros lógicos referenciados ou arquivos lógicos referenciados.

Nome do Caso de Uso: **Define Funções do Tipo Transação**
Categoria: Calcula Pontos de Função
Objetivo: Manter cadastrada as funções dos tipo transação pertinentes ao sistema em questão.

Fluxo Principal:
1 – usuário cadastra descrição da função
2 – usuário cadastra tipo de função (CE, SE ou EE)
3 - usuário cadastra dados elementares <<extend>> para cada função.
4 - sistema calcula dados elementares referenciados e registros lógicos referenciados para cada função

Fluxo Alternativo:
1 – se tipo de função (ação 2) é igual AIE definir se função é pertinente ou não ao aplicativo em questão.
   Se ação 1 não realizada
2 - usuário define funções referentes aos Arquivos Lógicos Internos
      - dados elementares referenciados
      - registros lógicos referenciados ou arquivos lógicos referenciados.

3 - usuário define funções referentes aos Arquivo de Interface Externa
      - dados elementares referenciados
      - registros lógicos referenciados ou arquivos lógicos referenciados.

Nome do Caso de Uso: **Cadastra dados elementares**
Categoria: Calcula Pontos de Função
Objetivo: Cadastrar os dados elementares pertinentes a cada função.
Fluxo Principal
1 - usuário cadastra dados elementar.
2 – usuário cadastra tipo de dado elementar (DER entrada, DER saída, RLR entrada, RLR saída).

Nome do Caso de Uso: **Relata Custo por Cargo**
Categoria: Emite Relatórios (projeto específico)
Objetivo: Demonstrar o custo médio de cada cargo por mês com relação ao presente projeto.

Fluxo Principal
1 - usuário solicita o gráfico
2 - o sistema busca as seguintes informações no banco de dados:
      - Tempo Total do projeto
      - Salário de cada funcionário
      - Cargo do Funcionário
      - Tempo Gasto no Projeto
      - Tempo de Trabalho por Mês de cada funcionário em horas
3 - Sistema calcula custo médio por cargo através da fórmula:
    Custo Cargo = Soma(Custo Funcionário)/Total Funcionários p/Cargo
4 - Sistema retorna as informações requeridas:
      - Cargo
      - Custo médio/mês
5 – o sistema gera o gráfico de barras

Fluxo Alternativo
1 - se necessário usuário imprime gráfico

Nome do Caso de Uso: **Relata o Esforço por Fase**
Categoria: Emite Relatórios (projeto específico)
Objetivo: Gráfico com o total de horas e percentual dedicados, em um certo período de tempo, em cada fase do ciclo de desenvolvimento de um certo sistema considerando todo o pessoal da área de informática.

Fluxo Principal
1 - usuário solicita o gráfico
2 - o sistema busca no banco de dados as seguintes informações para cada funcionário:
      - Horas Gastas em cada atividade
      - Fase da Atividade
3 - o sistema calcula o esforço da seguinte maneira para cada fase:
            Resultado = Soma(Horas Gastas)
4 - o sistema retorna as informações requeridas:
      - Fase
      - Total de Horas Gasta e Percentual
5 – o sistema gera o gráfico de barras

Fluxo Alternativo
1 - se necessário usuário imprime gráfico

Nome do Caso de Uso: **Relata o Perfil Profissional por Fase**
Categoria: Emite Relatórios (projeto específico)
Objetivo: Mostra a quantidade de horas X homens e seu respectivo percentual utilizados em cada fase do ciclo de desenvolvimento de um certo sistema, subdividido por tipo de perfil profissional (cargo).

Fluxo Principal
1 - usuário solicita o gráfico
2 - o sistema busca no banco de dados as seguintes informações para cada funcionário:
- Cargo do Funcionário
- Horas Gastas em cada atividade
- Fase da Atividade

3 - o sistema calcula o esforço da seguinte maneira para cada fase:
Resultado = Soma (Horas Gastas) classificados por cargo
4 - o sistema retorna as informações requeridas:
- cargo
- fase
- total de horas
- percentual

Fluxo Alternativo
1 - se necessário usuário imprime gráfico

Nome do Caso de Uso: **Relata Custo por Fase**
Categoria: Emite Relatórios (projeto específico)
Objetivo: Demonstrar os custos percentuais por cada fase do ciclo de desenvolvimento de sistemas, para uma aplicação específica.

Fluxo Principal
1 - usuário solicita o gráfico.
2 - o sistema busca no banco de dados as seguintes informações para cada funcionário: e os demais recursos??
- Salário
- Total de horas mês de trabalho
- Horas Gastas em cada atividade que desenvolveu
- Fase da Atividade

3 - o sistema calcula o custo para cada atividade:
Custo Atividade = (Salário/Total Horas Trabalho) * Horas Gastas
4 - o sistema agrupa o Custo Atividade para cada fase e faz o somatório do Custo Atividade, definindo assim o total de custos para cada fase.
5 - o sistema retorna as informações requeridas
- Fase
- Custo em % e R$

6 – o sistema gera o gráfico de barras

Fluxo Alternativo
1 - se necessário usuário imprime gráfico

Nome do Caso de Uso**: Apresenta Gráfico Alocação de Recursos**

Categoria: Emite Relatórios (projeto específico)
Objetivo: mostrar a quantidade de trabalho delegada a um determinado recurso por período de 1 dia. Levando em consideração o horário de trabalho deste recurso.

Fluxo Principal:
1 - usuário determina qual recurso será analisado pelo gráfico
2 - usuário solicita o gráfico
3 - o sistema busca no banco de dados as seguintes informações:
- nome funcionário
- atividades alocadas para este funcionário
- data de início da atividade
- data de término da atividade
- quantidade de recurso alocada para esta atividade
4 - sistema calcula " Alocação de Recursos " com base nas informações acima.
5 - sistema retorna as informações requeridas:
- nome do funcionário
- data
- % alocado por data

Fluxo Alternativo:
1 - se necessário usuário imprime o gráfico.

Nome do Caso de Uso: **Compara Tamanho do Sistema**
Categoria: Emite relatórios (comparativo entre projetos)
Objetivo: este indicador analisa os sistemas implantados classificando-os por tamanho em pontos de função.

Fluxo Principal:
1 - usuário solicita gráfico comparativo
2 - o sistema busca no banco de dados dos respectivos projetos em avaliação a quantidade de PF ajustados
3 - o sistema classifica o tamanho do projeto de acordo com a tabela a seguir:

Entre (PF)	
100 a 500	Pequeno
500 a 2000	Médio
2000 a 5000	Grande
5000 a 20000	Muito Grande
20000 a 50000	Gigante

4 - o sistema retorna as informações requeridas:
Informações gerais:
- Classificação
- Percentual

Informações detalhadas:
- Nome do projeto
- classificação (grande, médio,.....)
- quantidade de PF

5 – o sistema gera gráfico de pizza.

Fluxo Alternativo
1 - se necessário usuário imprime gráfico comparativo

Nome do Caso de Uso: **Compara Fatores de Ajustes**
Categoria: Emite relatórios (comparativo entre projetos)

Objetivo: Comparar entre vários projetos os fatores de ajustes das aplicações.

Fluxo Principal
1 - usuário solicita gráfico comparativo
2 - o sistema busca no banco de dados dos respectivos projetos em avaliação o valor dos fatores de ajustes de cada aplicação, que pode variar de 0,78 a 1,22.
3 - o sistema retorna as informações requeridas:
- nome do projeto
- Fator de ajuste
4 – sistema retorna o gráfico de linha.

Fluxo Alternativo
1 - se necessário usuário imprime gráfico comparativo

Nome do Caso de Uso: **Compara PF bruto por aplicação**
Categoria:Emite relatórios (comparativo entre projetos)
Objetivo: Este indicador mostra como se dividem os pontos de função das aplicações.

Fluxo Principal
1 - usuário solicita gráfico comparativo
2 - o sistema busca no banco de dados dos respectivos projetos em avaliação a quantidade de PF bruto de cada aplicação dividindo-os nos 5 tipos de funções existentes (Arquivo de Interface Externa, Arquivos lógicos Internos, Entradas Externas, Saídas Externas, Consultas Externas);
3 - o sistema retorna as informações requeridas:
- Nome do Projeto
- Pontos de Função classificados por:
    * Arquivo de Interface Externa
    * Arquivos Lógicos Internos
    * Entradas Externas
    * Saídas Externas
    * Consultas Externas
- Total de PF bruto de cada aplicação

Fluxo Alternativo
1 - se necessário usuário imprime gráfico comparativo

Nome do Caso de Uso: **Compara $/PF entre projetos**
Categoria: Emite relatórios (comparativo entre projetos)
Objetivo: este indicador mostra um comparativo de preços para se desenvolver um ponto de função nas diversas aplicações cadastradas.

Fluxo Principal
1 - usuário solicita gráfico comparativo
2 - o sistema busca no banco de dados dos respectivos projetos em avaliação o valor necessário, em R$, para desenvolvimento de 1 PF.
3 – o sistema calcula o valor através da seguinte fórmula:
Resultado = Valor Total do Projeto (R$) / Soma total de PF ajustados
4 - o sistema retorna as informações requeridas:
- Nome do projeto
- Valor para desenvolvimento de 1 PF (em R$)
5 – o sistema gera o gráfico de barras

Fluxo Alternativo
1 - se necessário usuário imprime gráfico comparativo

Nome do Caso de Uso: **Compara Tempo/PF entre projetos**
Categoria: Emite relatórios (comparativo entre projetos)
Objetivo: este indicador mostra um comparativo do tempo de desenvolvimento de um ponto de função nas diversas aplicações cadastradas.

Fluxo Principal
1 - usuário solicita gráfico comparativo
2 - o sistema busca no banco de dados dos respectivos projetos em avaliação, o valor de tempo necessário para desenvolvimento de 1 PF
3 – o sistema calcula o valor através da seguinte fórmula:
Resultado = Tempo Total Desenvolvimento / Soma Total de PF ajustados
4 - o sistema retorna as informações requeridas
- Nome do Projeto
- Tempo para desenvolvimento de 1 PF
5 – o sistema retorna o gráfico de barras

Fluxo Alternativo
1 - se necessário usuário imprime gráfico comparativo

Use Case name: **Configura opções agente**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: manter em arquivo denominado REG.gmt as opções de configuração do agente Gemetrics, para futura utilização destas informações.

Fluxo Principal
1 - usuário define informações do tipo:
* reenvio automático de mensagens (Ativar/Desativar)
* tempo para reenvio automático de mensagens (em dias)
* tempo limite de histórico de mensagens (em dias)
2 - usuário gravas as informações em arquivo de configuração

Fluxo Alternativo
1 - usuário pode cancelar alteração de configuração

Use Case name: **Reenvia Mensagem**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: o agente local reenvia as mensagens que não foram confirmadas pelos agentes distribuídos.

Fluxo Principal
1 - usuário solicita o reenvio de mensagens
2 - agente verifica no arquivo log.gmt quais mensagens não foram enviadas.
3 - agente associa o registro do arquivo de log e o registro contido no arquivo mensagens.gmt.
4 - o agente consulta o arquivo dis?.gmt para localizar o endereço IP do agente distribuído que receberá a mensagem
5 - o agente envia a mensagem para o agente distribuído.

6 - Após enviar mensagem ficar no aguardo de confirmação de envio de mensagem - <use> Monitora recebimento de mensagem

Fluxo alternativo
1 - se o tipo de operador da mensagem for 1 o agente promove atualizações nos arquivos de configuração. <use> Mantem arquivo distribuído
2 - se o tipo de operador da mensagem for 3 o agente promove atualizações nos arquivos de configuração. <use> Atualiza Endereço IP

Use Case name: **Mantém arquivo distribuído**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: manter em arquivo do tipo DIS?.gmt as opções de identificação dos agentes distribuídos do agente Gemetrics, para futura utilização destas informações.

Fluxo Principal
1 - usuário define qual aplicativo está em sendo mantido.
2 - usuário define endereço IP
3 - usuário define Identificador do aplicativo local
4 - usuário grava informações no arquivo.
5 - mensagem é enviada para agente distribuído
6 - aguarda confirmação da mensagem com Identificador do aplicativo distribuído.

Fluxo Alternativo
1 - se mensagem da ação 6 não retornada por agente distribuído, ação 5 será reenviada após um ciclo de tempo.

Use Case name: **Monitora recebimento de mensagem**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: monitorar todas as mensagens recebidas pelo agente local enviadas pelos agentes distribuídos, tomando determinadas ações dependendo do tipo da mensagem recebida.

Fluxo Principal
1 - agente recebe mensagem
2 - agente identifica o tipo da mensagem para tomar determinada ação. (A especificação de cada mensagem está detalhada em documento específico.)

Fluxo alternativo
1 - se o tipo de operador da mensagem for 1 o agente promove atualizações nos arquivos de configuração. <use> Mantem arquivo distribuído
2 - se o tipo de operador da mensagem for 3 o agente promove atualizações nos arquivos de configuração. <use> Atualiza Endereço IP

Use Case name: **Atualiza Endereço IP**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: manter nos arquivos de configuração o endereço IP atual da máquina.

Fluxo Principal
1 - usuário solicita atualização do endereço IP.

2 - agente atualiza arquivo de mensagens.
3 - agente atualiza arquivos dis?.gmt
4 - agente envia mensagem ao agente distribuído a fim de atualizar os arquivos de mensagem e Dis?.gmt dos agentes distribuídos.

Fluxo Alternativo
1 - se mensagem for recebida de um processo distribuído executar ações 2 e 3 e enviar confirmação de recebimento de mensagem

---

Use Case name: **Solicita atualização para Gráfico Gantt**
Categoria: Gerenciamento Distribuído (agente)
Objetivo :Atualizar o projeto local com as informações referentes ao projeto correlato.

Fluxo Principal:
1 - solicitar data da última atualização do agente distribuído.
2 - sistema retorna data da última atualização
3 - Se data da última atualização do agente distribuído for maior que a última atualização do agente local, use <monitora BD Gemetrics> para solicitar atualização de dados

Fluxo Alternativo
1 - se ação 3 for negativa atualização é cancelada.

Use Case name: **Monitora BD Gemetrics**
Categoria: Gerenciamento Distribuído (agente)
Objetivo: monitorar alterações no BD Gemetrics a fim de manter a consistência das informações e viabilizar o gerenciamento distribuído

Pré-condição: receber sinal para envio de informação do aplicativo gemetrics

Fluxo Principal
1 - consultar dis?.gmt para recuperar IP do agente distribuído
2 - verifica detalhes da alteração ocorrida no BD
3 - envia mensagem
4 - agente armazena informações nos arquivos Dis?.gmt, mensag.gmt e log.gmt, quando necessário.

Fluxo alternativo
1 - se na ação 3 ocorrer problemas, nova mensagem será enviada posteriormente ao agente distribuído.

Obs.: Na especificação dos casos de uso referentes ao agente para o gerenciamento distribuído foram citados diversos arquivos que se farão necessários para fins de manutenção permanente de informações que serão utilizadas freqüentemente pelo agente. O agente Gemetrics trabalhará basicamente com 4 tipos de arquivos diferentes, são eles:

ü dis?.gmt – são os arquivos que serão usados para guardar informações imprescindíveis para localização do aplicativo certo no agente distribuído certo, ou seja, este arquivo conterá o Identificador do aplicativo Local e os Identificadores dos aplicativos distribuídos e seus respectivos endereços para localização na rede (Endereço IP ou host), informações utilizadas para futuras transferências de

informações. ? – significa o Identificador do aplicativo local, pois cada aplicativo local terá um arquivo Dis?.gmt;

ü mensag.gmt – este arquivo conterá as mensagens enviadas pelo agente local aos demais agentes, essas informações são armazenadas para futuros reenvio de mensagens, se houver necessidade.

ü log.gmt – este arquivo conterá a identificação de cada mensagem enviada pelo agente local aos agentes distribuídos e a confirmação ou não do recebimento destas mensagens por parte dos agentes distribuídos. Mensagens não enviadas corretamente aos agentes distribuídos, por qualquer que seja o motivo, serão posteriormente enviadas com base nas informações armazenadas neste local.

ü reg.gmt – este arquivo contém informações de configuração do agente, tais como:

* Ativar/Desativar reenvio automático de informações

* Tempo para reenvio de informações

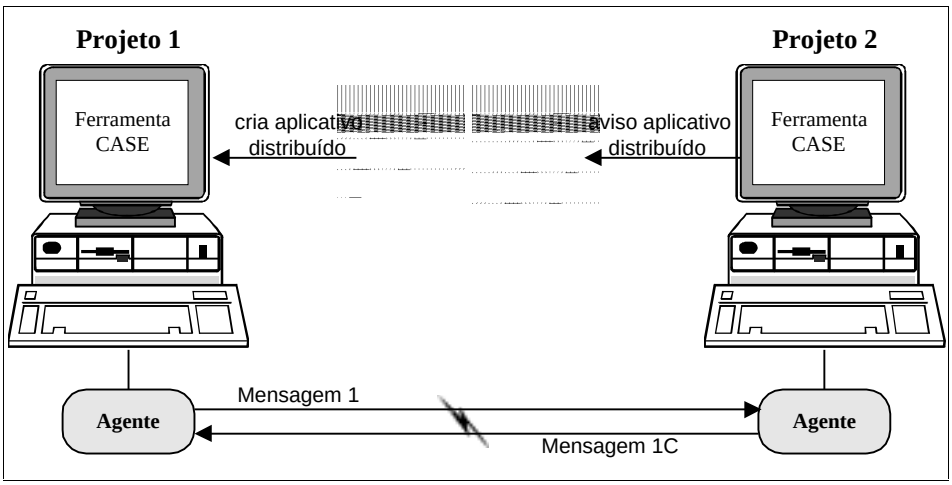
* Número de dias que as informações no histórico de mensagens.

Apêndice B

Especificação das ações do agente e
das mensagens trocadas entre os agentes

Ação 1: Criação de aplicativo distribuído

No momento da criação de um aplicativo distribuído o agente é ativado solicitando o cadastramento dos endereços IP dos agentes responsáveis pelos projetos correlatos. E, a partir desta informação o agente informa aos agentes correlatos, conforme figura abaixo, a criação de um aplicativo distribuído.



MENSAGEM 1

Mensagem utilizada para criar o aplicativo na máquina distribuída e o arquivo de configuração (dis?.gmt) do aplicativo distribuído, onde ? é o ID do aplicativo a ser criado.

1 – Id_msg	2 – Id_operação	3 - Host	4 – ID aplicativo Local	5 – Aplic. descrição

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 1.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo criado na máquina local.
5 – Descrição do aplicativo criado na máquina local.

MENSAGEM 1C

Mensagem utilizada pelo agente distribuído para confirmar o recebimento da mensagem anterior (mensagem 1).

1 – Id_msg	2 – Id_operação	3 - Host	4 – ID aplicativo Local	5 – Id_msg_conf

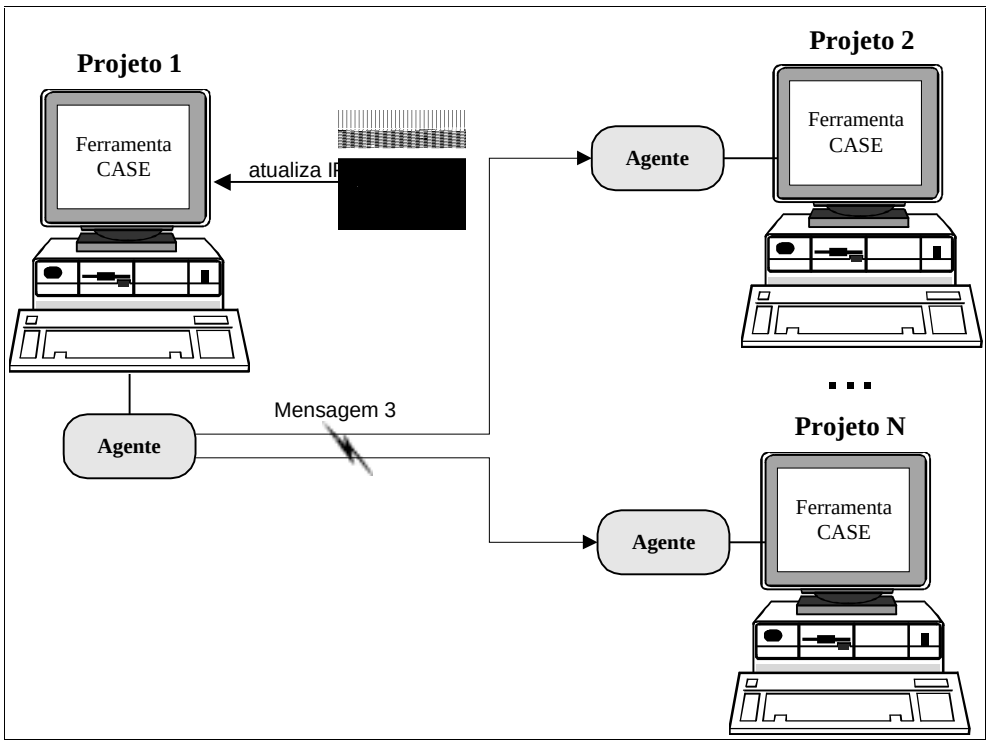
1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 1C.

3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo criado na máquina local.
5 – Identificador da mensagem a ser respondida, enviada anteriormente.

Ao receber esta mensagem, a máquina local, atualiza o arquivo de log (log.gmt) de acordo com o Id_msg_conf, indicando que a mensagem 1 enviada anteriormente por ele foi recebida com sucesso pela máquina distribuída. Além disso, a máquina local atualizará o seu arquivo de configuração, associando o identificador do aplicativo local com o recebido na mensagem.

Ação 2: Atualização de endereço do agente

Caso ocorra mudança no endereço IP de algum dos agentes a mensagem 3 é utilizada para comunicar a alteração a todos os agentes correlatos, conforme figura abaixo.



MENSAGEM 3

Mensagem utilizada pelo agente com o propósito de modificar o endereço IP do arquivo de configuração (dis?.gmt) das máquinas distribuídas que se relacionem com a máquina local.

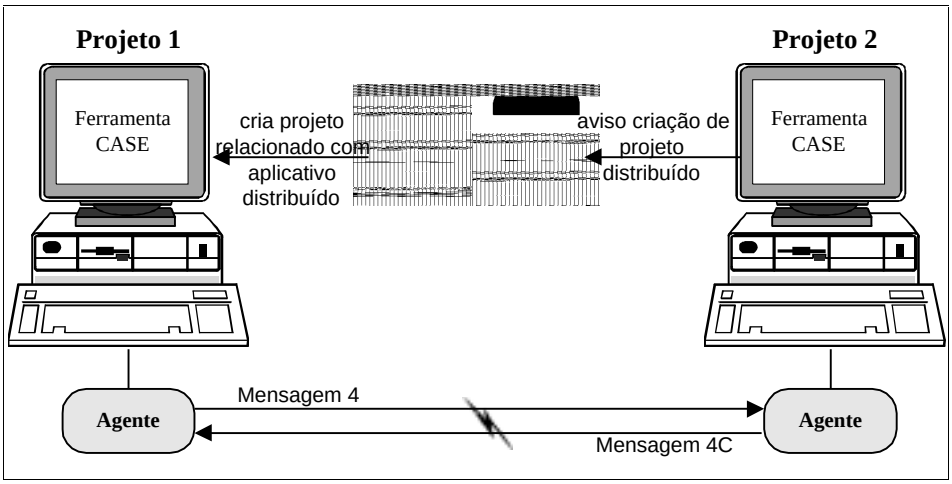
1 – Id_msg	2 – Id_operação	3 – Host	4 – ID aplicativo D

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 3.

3 – Endereço IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é recuperada nos arquivos de configurações (dis?.gmt) de cada aplicativo.

Ação 3: Criação de projeto correlato

As mensagens 4 e 4C são utilizadas para viabilizar a criação dos projetos correlatos, conforme figura abaixo.



MENSAGEM 4

No momento de inserir um novo projeto no aplicativo Gemetrics, e este for vinculado a um aplicativo distribuído, o agente utilizará a Mensagem 4, sendo esta, utilizada pelo agente com o propósito de criar o projeto em questão no agente distribuído.

1 – Id_msg	2 - Id_operação	3 - Host	4 – ID aplicativo D	5 – Descrição
6 – Data Início	7 - anotações	8 - autor	9 - assunto	10 - empresa

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 4.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é recuperada dos arquivos de configurações (dis?.gmt) de cada aplicativo.
5 – Descrição do projeto em questão, ou seja, o nome do projeto.
6 – Data de início do projeto.
7 – Anotações sobre o projeto.
8 – Autor do projeto.
9 – Assunto do projeto.
10 – Empresa responsável pelo projeto.

MENSAGEM 4C

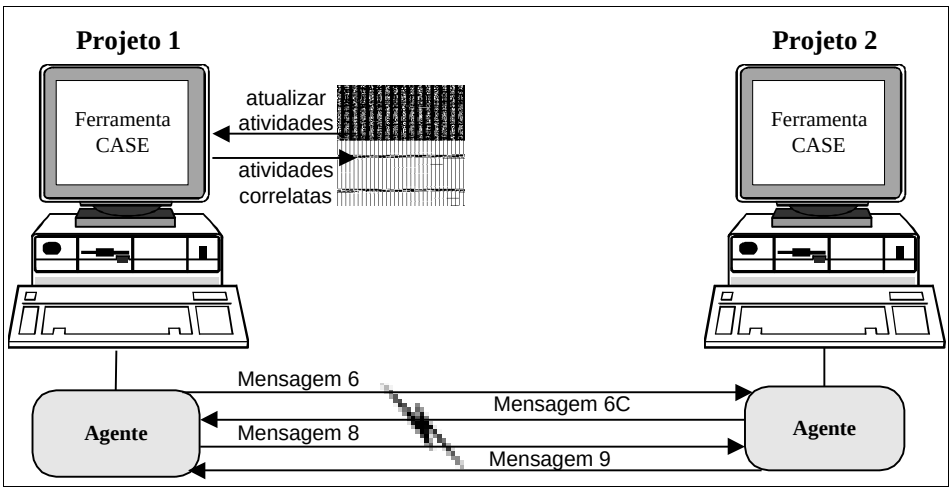
Mensagem utilizada pelo agente com o propósito de confirmar o recebimento e sucesso na operação efetuada pela mensagem 4.

1 – Id_msg	2 - Id_operação	3 - Host	4 – Id_msg_conf

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 3.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador da mensagem recebida na mensagem com Id_operação 4, enviada anteriormente.

Ação 4: Atualização das atividades correlatas

Quando solicitada a atualização das atividades, para o gerente obter uma visão global, ocorre a seqüência e mensagens constantes da figura abaixo.



MENSAGEM 6

Quando o usuário requer “atualizar atividade” no aplicativo Gemetrics, o agente Local deve solicitar ao agente distribuído, todas as atividades do projeto vinculado ao aplicativo ora em questão. Para solicitar ao agente distribuído as atividades o agente local deverá utilizar a mensagem 6.

1 – Id_msg	2 - Id_operação	3 – Host	4 – ID aplicativo D

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 6.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é recuperada nos arquivos de configurações (dis?.gmt) de cada aplicativo.

MENSAGEM 6C

A mensagem 6C tem o propósito de confirmar o recebimento da mensagem 6 e informar a quantidade de atividades e sub-atividades que serão enviadas por ele, cabendo ao agente que solicitou as informações confirmar ou não o download das atividades e sub-atividades.

1 – Id_msg	2 - Id_operação	3 - Host	4 – ID aplicativo D	5 – Qtd. Atividades
6 – Qtd. Sub-ativid.	7 - Id_msg_conf			

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 6C.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é recuperada nos arquivos de configurações (dis?.gmt) de cada aplicativo.
5 – Quantidades de atividades que serão enviadas pelo agente local.
6 – Quantidades de sub-atividades que serão enviadas pelo agente local.
7 – Identificador da mensagem da 6, enviada anteriormente.

MENSAGEM 8

Ao receber a mensagem 6C, o usuário será informado do número de atividades e sub-atividades envolvidas na transação e será questionado sobre o envio ou não destas informações. Se o usuário optar pelo recebimento das informações, ele deverá utilizar a mensagem 8 para confirmar o início da transação.

1 – Id_msg	2 - Id_operação	3 - Host	4 – ID aplicativo D

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 8.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é recuperada nos arquivos de configurações (dis?.gmt) de cada aplicativo.

MENSAGEM 9

A mensagem 9 transportará as informações referentes às atividades e as sub-atividades e os links que serão passadas ao agente distribuído. Cada projeto utilizará uma mensagem 9, ou seja, todas as atividades, sub-atividades e links para serem transportadas, serão enviadas nesta mensagem ao agente solicitante.

1 – Id_msg	2 - Id_operação	3 – Host Local	4 – ID aplicativo D	5 – Host Distribuído	6 – Arquivo em Anexo

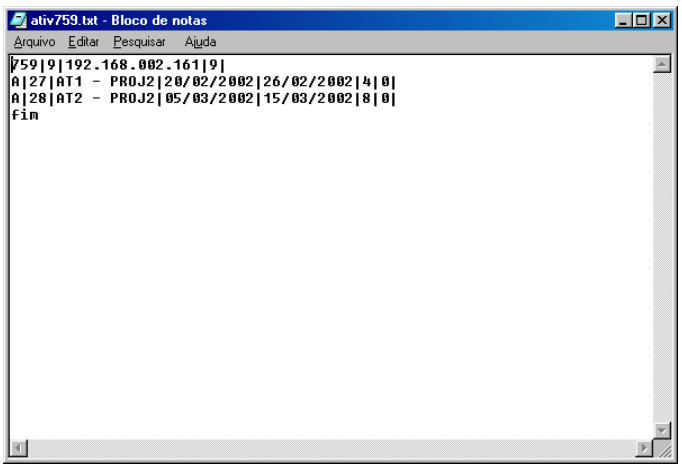
1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 9.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é buscada nos arquivos de configurações (dis?.gmt) de cada aplicativo.
5 – Endereço IP da máquina que recebe a mensagem – IP distribuído.
6 - Arquivo em Anexo - segue a especificação a seguir:
SE for Atividade (formato da linha)
    1 – 'A' (de Atividade)
    2 - Código identificador da atividade ora em questão.
    3 – Descrição da atividade
    4 – Data de início da atividade.
    5 – Data final da atividade
    6 – Número de dias da atividade
    7 – Percentual de conclusão da atividade (0-100%).
SE for Sub-atividade (formato da linha)
    1 – 'S' (de Sub-Atividade)
    2 - Código identificador da sub-atividade ora em questão.
    3 - Código identificador da atividade
    5 – Descrição da sub-atividade
    4 – Data de início da sub-atividade.
    5 – Data final da sub-atividade
    6 – Número de dias da sub-atividade
    7 – Percentual de conclusão da sub-atividade (0-100%).
SE for Link (formato da linha)
    1 – 'L' (de Links)
    2 – Ativ_código/ Ativ_espelho
    3 - Atividade Predecessora (*)
    4 – Tipo de Link (L para Local/ D para Distribuído)
(*) Código da atividade com a qual esta possui dependência (link). Se o tipo do link for local será enviado o Ativ_codigo da atividade; mas se for Distribuído será enviado o Ativ_espelho da atividade ora em questão.

A tabela atividade contém um campo denominado Ativ_espelho o qual contém o ativ_código do projeto de origem da atividade. Desta forma, somente as atividades que não são locais possuem este campo preenchido.
Para cada atividade da mensagem 9 recebida fazer:
      Verificar se ativ_espelho = ativ_código se existir
            Atualiza registro
            Marcar como atual
      Senão
            Inserir registro
            Marcar como atual
Após receber a última mensagem o agente irá deletar todos os registros de atividades distribuídas não marcadas como "atual".

Para cada Link da mensagem 9 recebida fazer:
      Se TipoLink=D fazer
            Verificar ativ_código = Pred_ativ_código (campo 11)
            Pred_ativ_código (local) = Pred_ativ_código (campo 11)
      Se TipoLink=L fazer
            Verificar ativ_espelho = Pred_ativ_código (campo 11)
            Pred_ativ_código (local) = Pred_ativ_código (campo 11)
      Gravar na tabela Links

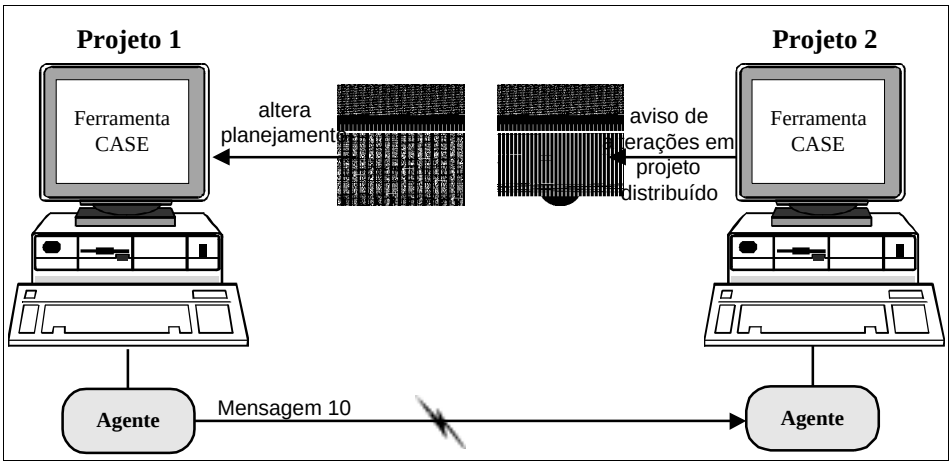
Para cada sub-atividade da mensagem 9 recebida fazer:
Verificar se ativ_espelho = ativ_código (campo6) se existir
Ativ_código (campo6) = Ativ_código (local)
Gravar sub-atividade

Exemplo de um arquivo da mensagem 9 com apenas 2 atividades.



Ação 5: Alteração do Planejamento

Quando um dos gerentes altera o planejamento inicial e esta alteração interfere em projetos correlatos, ocorre a seqüência e mensagens da figura abaixo.



MENSAGEM 10

A mensagem 10 transportará as informações (data inicial e final) referente as atividades as sub-atividades que serão passadas para o agente distribuído. Isto ocorrerá quando a otimização (PERT) for executada num aplicativo local e essas atividade estejam vinculadas a um projeto distribuído.

1 – Id_msg	2 - Id_operação	3 – Host Local	4 – ID aplicativo D	5 – Host Distribuído	6 – Arquivo em Anexo

1 – Identificador da mensagem, ou seja, o número que identificará esta mensagem no arquivo de mensagens (mensag.gmt)
2 – Identificador da operação, ou seja, indicará ao agente distribuído qual operação ele deve executar ao receber esta mensagem, neste caso o valor da operação é 9.
3 – Endereço de IP da máquina que envia a mensagem – IP local.
4 – Identificador do aplicativo distribuído, esta informação é buscada nos arquivos de configurações (dis?.gmt) de cada aplicativo.
5 - Endereço de IP da máquina que recebe a mensagem – IP Distribuído.
6 - Arquivo em Anexo
SE for Atividade (formato da linha)
  1 – Ativ_Espelho (Código da Atividade no projeto Distribuído)
  2 – Data Inicial
  3 – Data Final

Mensagens de Comunicação Interna

Estas mensagens são utilizadas para comunicação entre o Aplicativo Gemetrics (Ferramenta CASE) e o agente local.

MENSAGEM ATUALIZAR ATIVIDADE

Ocorre quando executa-se o PERT localmente e mudanças nas atividades distribuídas são detectadas.

‘atualiza_atividades’	2 – Cód_aplicativo

MENSAGEM CRIAR APLICATIVO

Ocorre quando um novo aplicativo distribuído é criado na ferramenta Gemetrics.

‘criar_aplicativo’	2 – Cód_aplicativo	3 – Aplic_descrição

MENSAGEM CRIAR PROJETO

Ocorre quando um novo projeto que pertence a um aplicativo distribuído é criado na ferramenta Gemetrics.

‘criar_projeto’	2–Cód_aplicativo	3–Cód_Projeto	4–Projeto_descrição	5–Projeto_inicio
6–Projeto_anotações	7–Projeto_Autor	8–Projeto_Assunto	9–Projeto_Empresa	

MENSAGEM BUSCAR ATIVIDADES

Ocorre quando é requerida a atualização de atividades no aplicativo Gemetrics.

‘buscar_atividades’	2 – Cód_aplicativo

ANEXOS

Esta seção apresenta os anexos que se fazem necessários para um maior detalhamento do trabalho. Assim, são apresentados os seguintes anexos:

Anexo 1: Questionário aplicado para estabelecimento dos requisitos.

Anexo 2: Telas da Ferramenta CASE GEMETRICS.

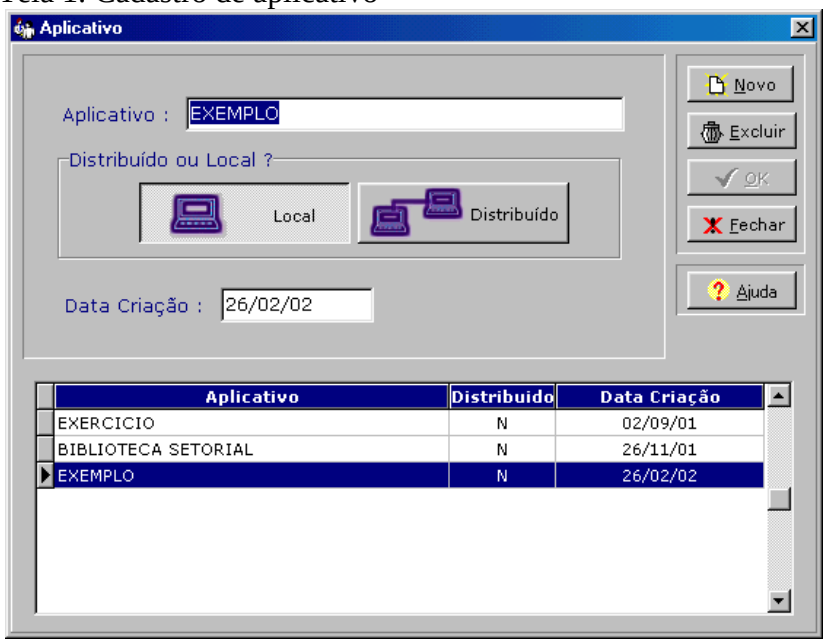
ANEXO 1

Questionário Aplicado para Levantamento de Requisitos junto a Empresas Desenvolvedoras de Software

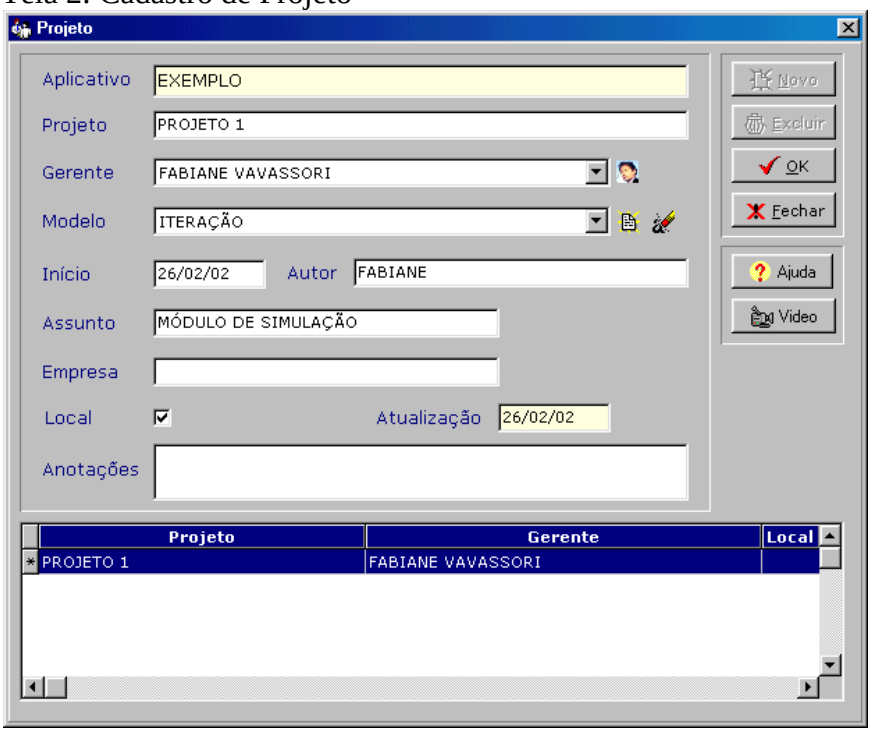
ANEXO 2

Telas da Ferramenta CASE GEMETRICS

Tela 1: Cadastro de aplicativo

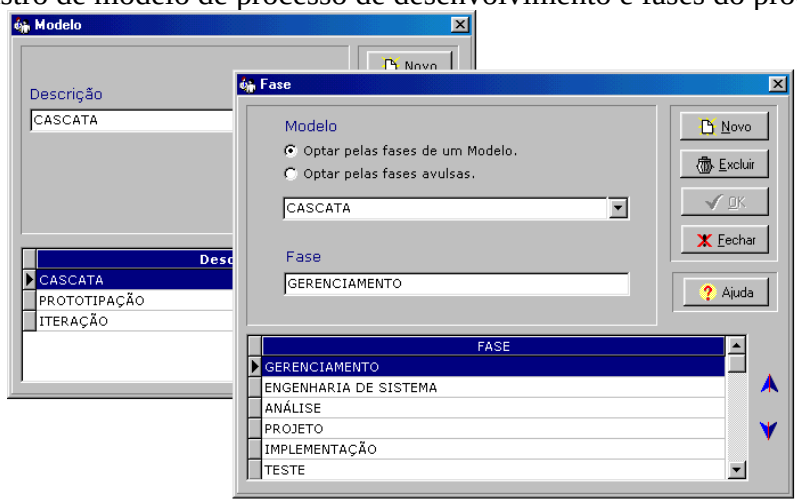


Tela 2: Cadastro de Projeto

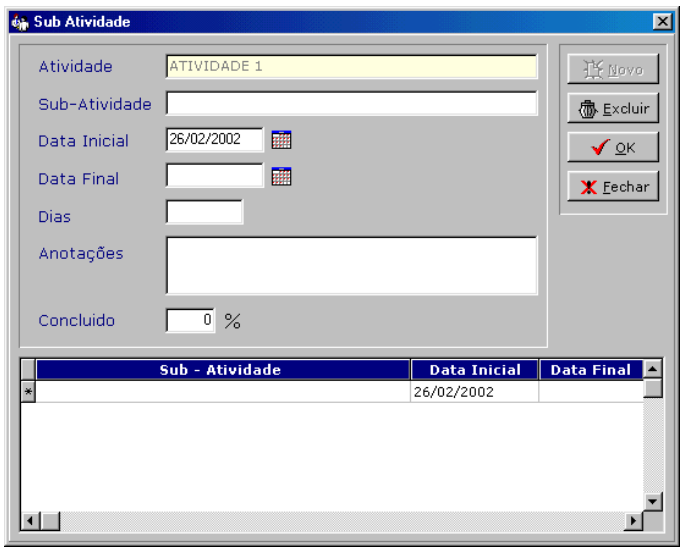


Tela 3: Cadastro de Gerente

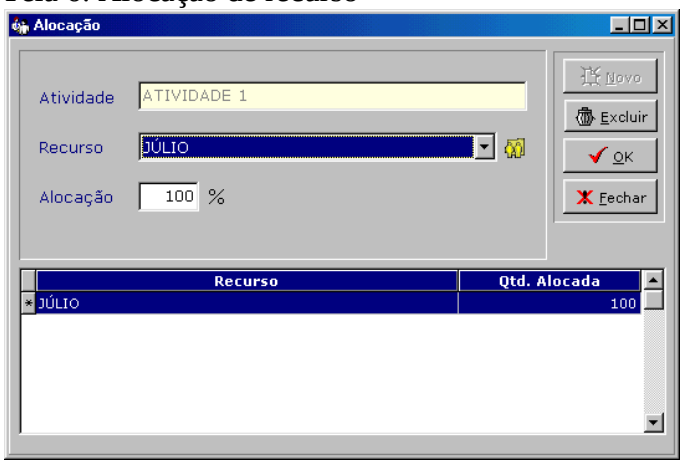
Tela 4: Cadastro de modelo de processo de desenvolvimento e fases do processo



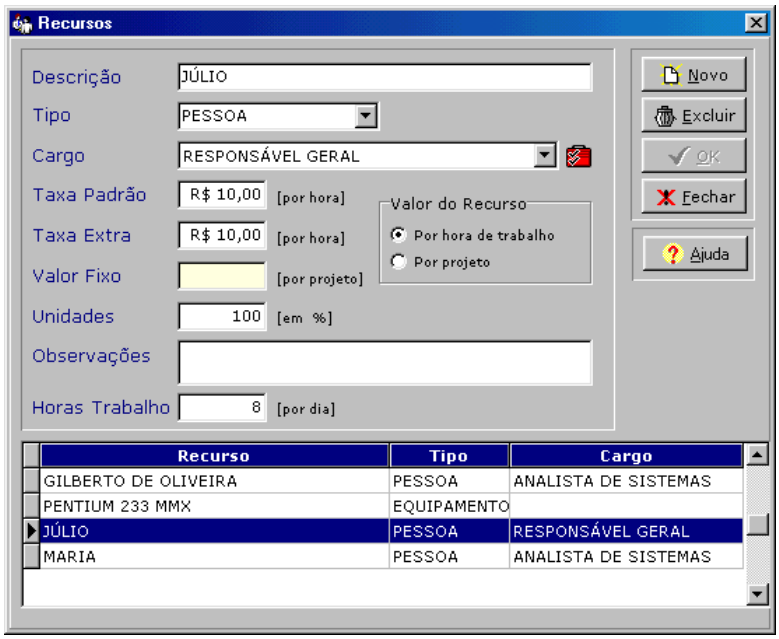
Tela 5: Cadastro de sub-atividade



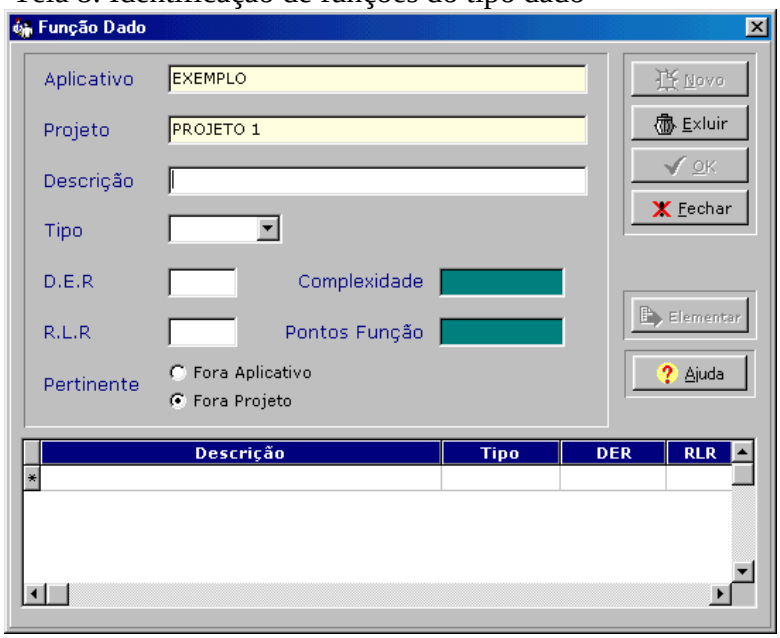
Tela 6: Alocação de recurso



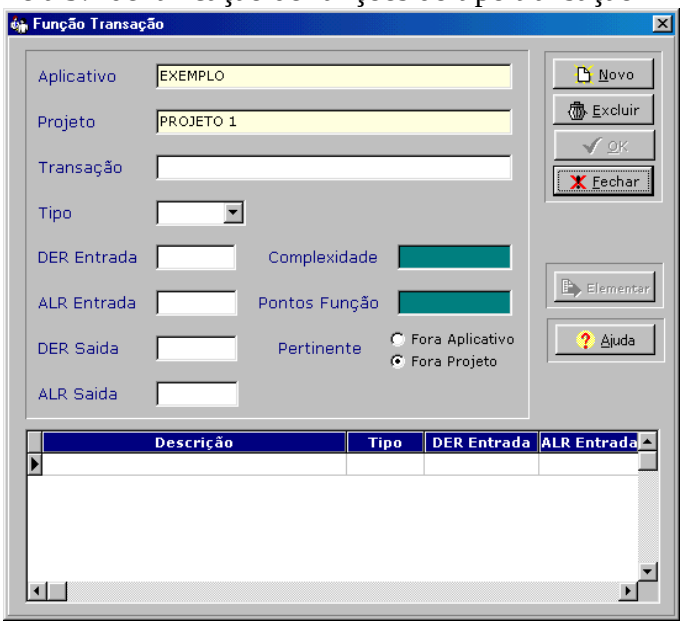
Tela 7: Cadastro de recurso



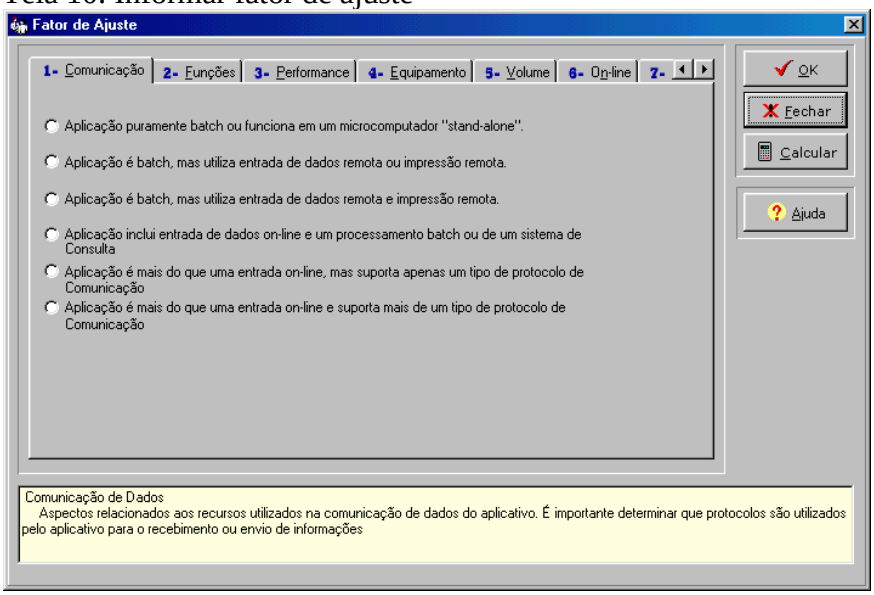
Tela 8: Identificação de funções do tipo dado



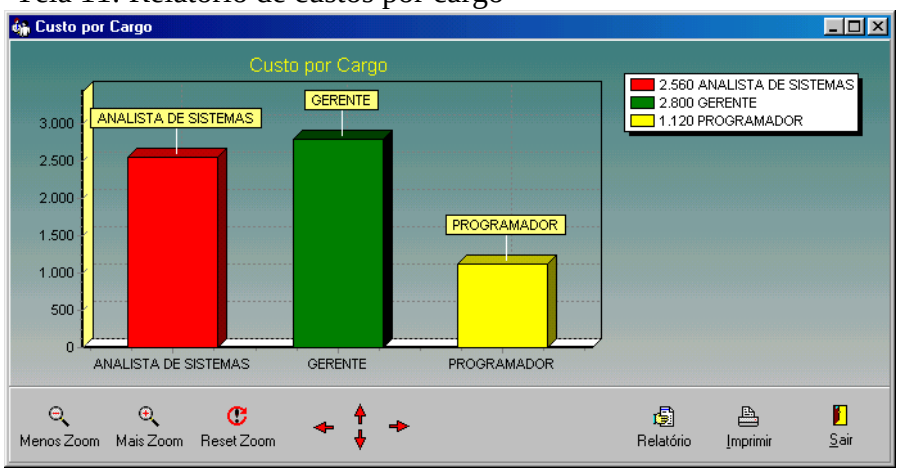
Tela 9: Identificação de funções do tipo transação



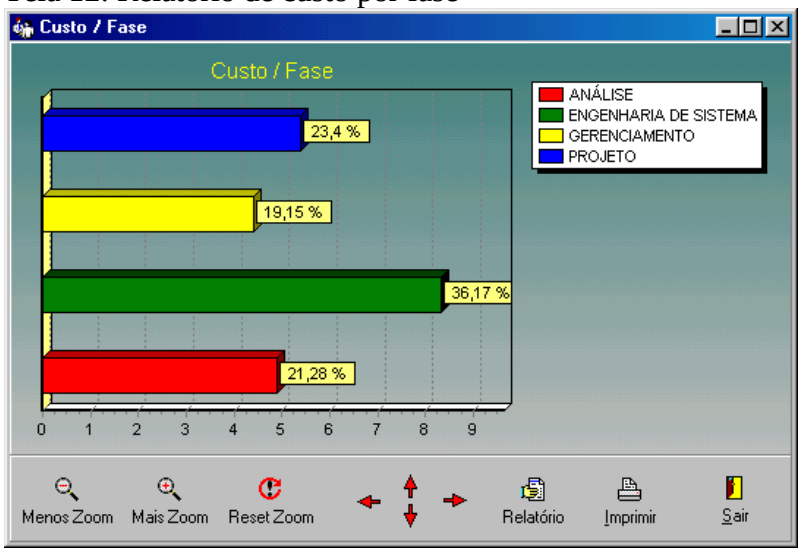
Tela 10: Informar fator de ajuste



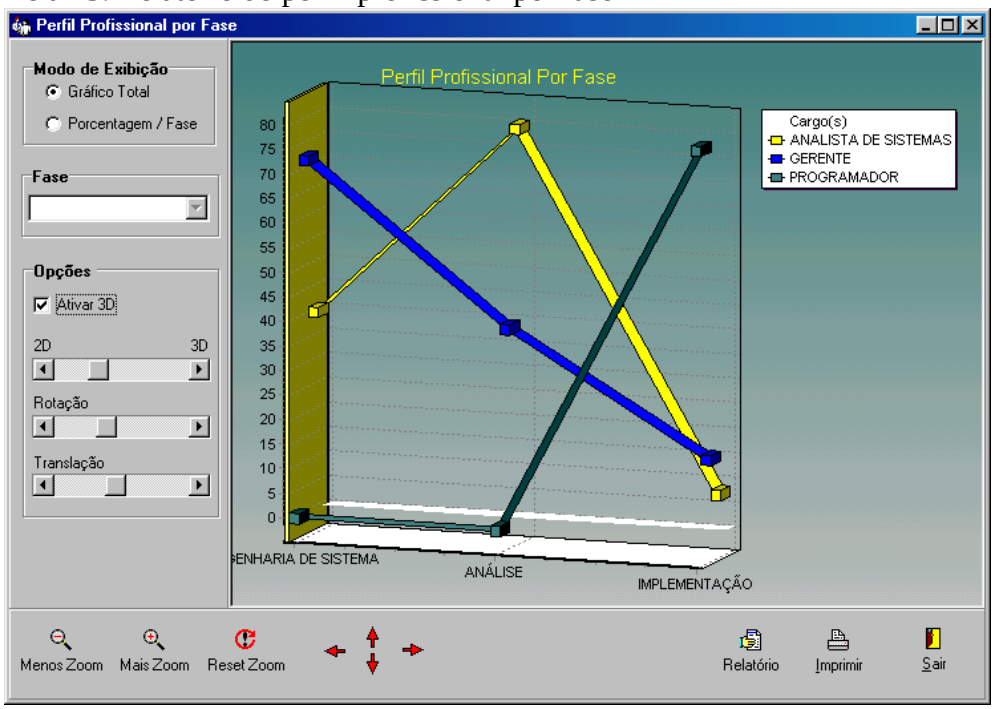
Tela 11: Relatório de custos por cargo



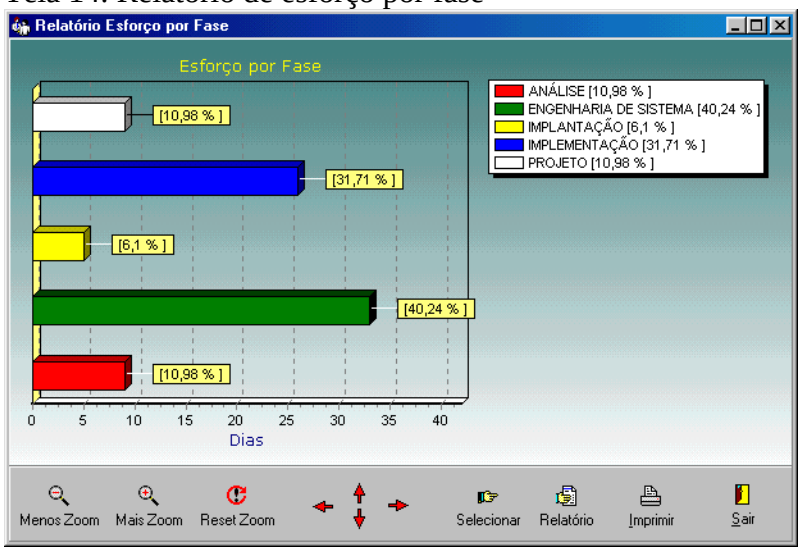
Tela 12: Relatório de custo por fase



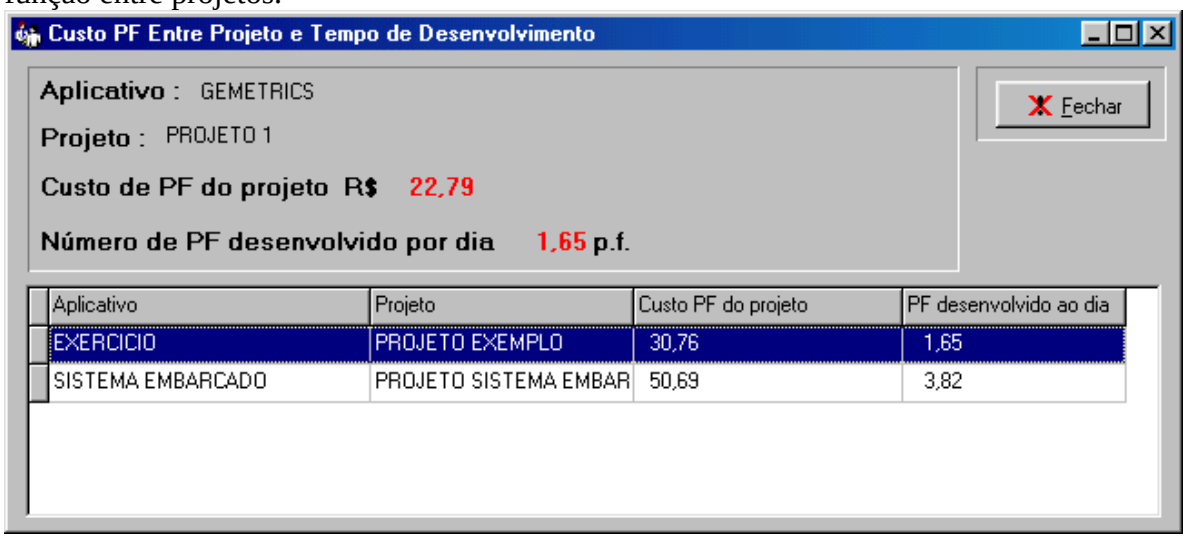
Tela 13: Relatório do perfil profissional por fase



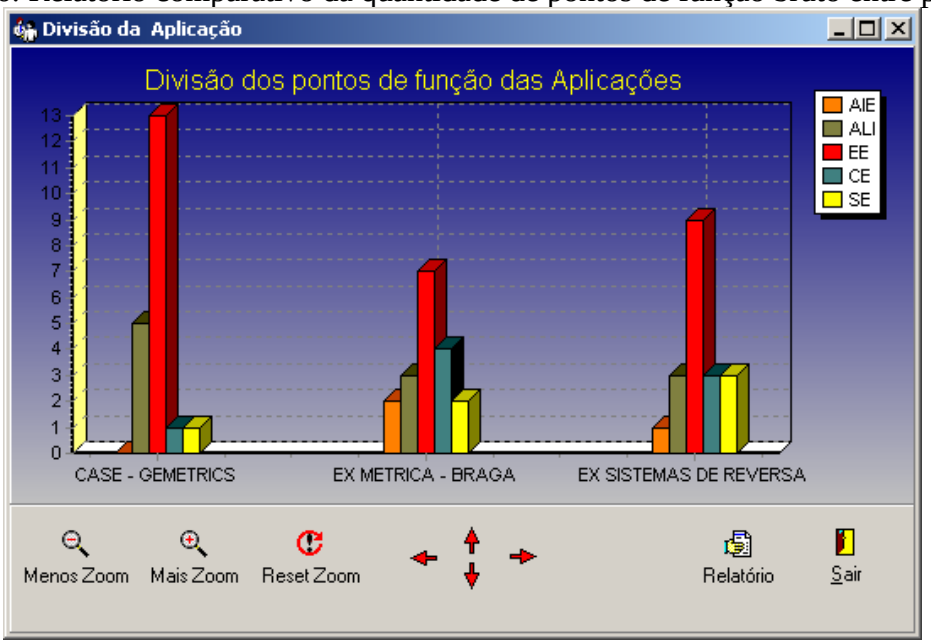
Tela 14: Relatório de esforço por fase



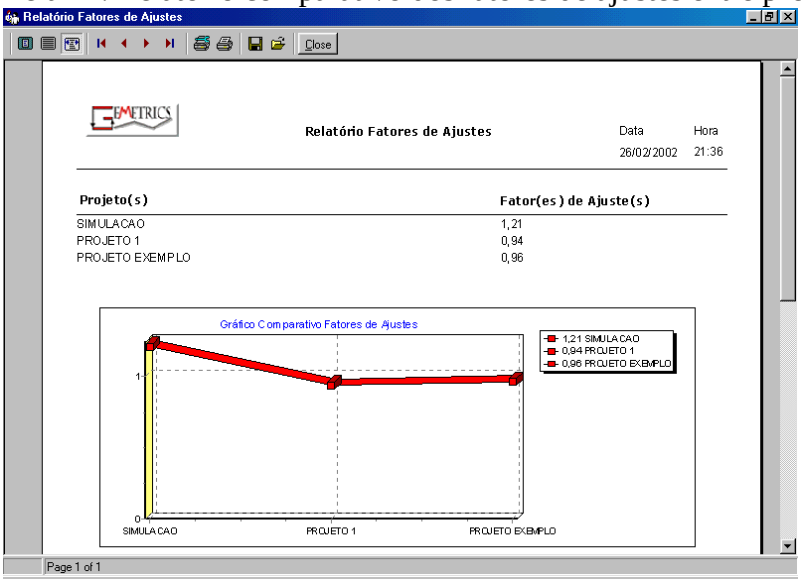
Tela 15: Relatório comparativo do tempo e custo de desenvolvimento de um ponto de função entre projetos.



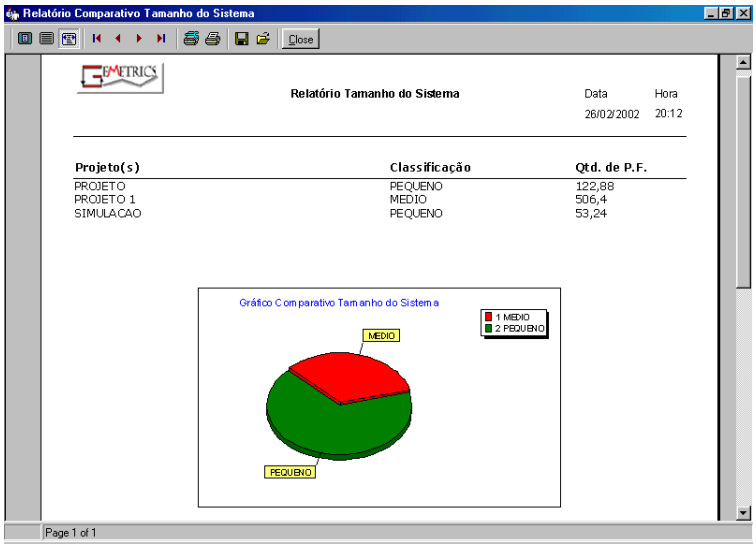
Tela 16: Relatório comparativo da quantidade de pontos de função bruto entre projetos



Tela 17: Relatório comparativo dos fatores de ajustes entre projetos

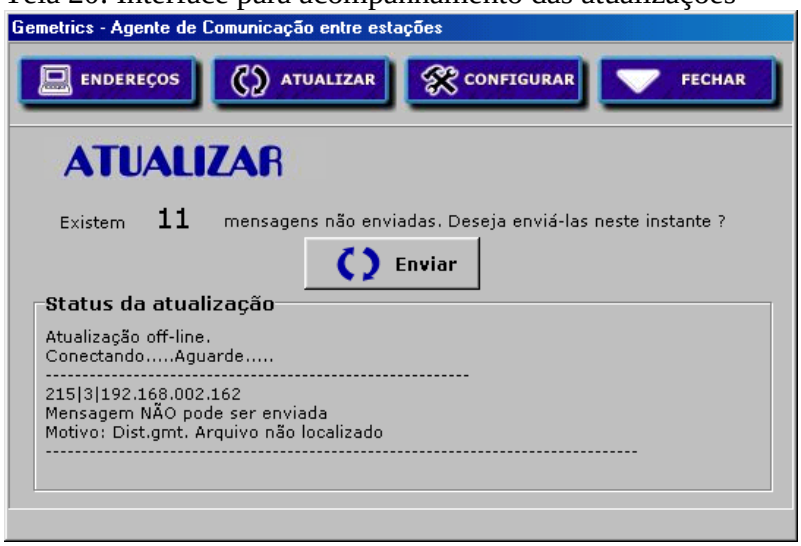


Tela 18: Relatório referente ao tamanho dos sistemas



Tela 19: Estimativas do Aplicativo

Tela 20: Interface para acompanhamento das atualizações



Tela 21: Interface para configurações